SCALABLE ONLINE DECENTRALIZED SMOOTHING AND MAPPING

A Dissertation
Presented to
The Academic Faculty

by

Alexander G. Cunningham

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2014

SCALABLE ONLINE DECENTRALIZED SMOOTHING AND MAPPING

Approved by:

Dr. Frank Dellaert, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Henrik Christensen
School of Interactive Computing
Georgia Institute of Technology

Dr. Ayanna Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Tucker Balch
School of Interactive Computing
Georgia Institute of Technology

Dr. Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Stergios Roumeliotis
Computer Science and Engineering
University of Minnesota

Date Approved: 2 April 2014

ACKNOWLEDGEMENTS

First and foremost, I thank my advisor Dr. Frank Dellaert for his support and guidance during my Ph.D. Dr. Dellaert worked tirelessly to provide a supportive academic environment, and pushed me to become a better researcher, and for that I cannot thank him enough. I would also like to thank my co-advisor Dr. Ayanna Howard, for providing counsel throughout my time at Georgia Tech. I would like to thank my Ph.D. committee Dr. Magnus Egerstedt, Dr. Henrik Christensen, Dr. Tucker Balch and Dr. Stergios Roumeliotis, as well. My work would also not have been possible without the funding and support of the Micro Autonomous Systems and Technology (MAST) project, sponsored through the Army Research Labs.

My current and past colleagues throughout both the BORG lab and through the Center for Robotics and Intelligent Machines were invaluable in making my work possible, from developing the software and math that forms the basis of this work, to conducting robot experiments. In addition, many of my colleagues at RIM are some of my closest friends, and I thank you all for keeping me sane during all of the late nights spent in the lab.

Finally, I want to thank my family for always being supportive throughout my years as a graduate student, and ensuring that I saw through this work to its completion.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

Many applications for field robots can benefit from large numbers of robots, especially applications where robots are trying to cover or explore a region. A key enabling technology for robust autonomy in these teams of small and cheap robots is the development of collaborative perception to account for the shortcomings of the small and cheap sensors on the robots.

In this dissertation, I present DDF-SAM to address the decentralized data fusion (DDF) problem with a decentralized inference based on the smoothing and mapping (SAM) approach to single-robot mapping that is online, scalable and consistent while supporting a variety of sensing modalities. The DDF-SAM approach performs fully decentralized simultaneous localization and mapping through a process in which robots choose a relevant subset of variables from their local map to share with neighbors. Each robot summarizes their local map to yield a density on exactly this chosen set of variables, and then distributes this *summarized map* to neighboring robots, which then further distribute the summarized map throughout the network. Each robot fuses summarized maps it receives to extend its local sensor horizon.

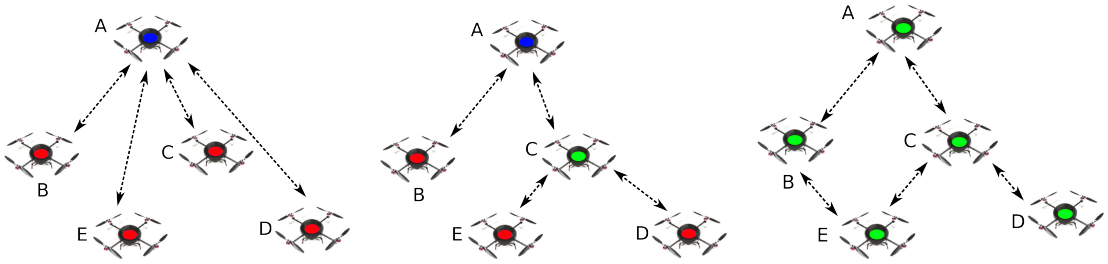
I introduce two primary variations on DDF-SAM, one that uses a batch nonlinear constrained optimization procedure to share maps, DDF-SAM 1.0, and one that uses an incremental solving approach for substantially faster performance, DDF-SAM 2.0. I validate these systems using a combination of real-world and simulated experiments. In addition, I evaluate design trade-offs for operations within DDF-SAM, with a focus on efficient approximate map summarization.

Chapter 1

INTRODUCTION

Many applications for field robots can benefit from large numbers of robots, especially applications where robots are trying to cover or explore a region. Scaling a robot application to teams of robots, particularly as the teams increase in size, is a problem where autonomy becomes critical to mitigate the need for human operators to manually control each robot separately. Some example applications where autonomous multi-robot teams are an improvement on single-robot solutions includes mobile sensor networks for environmental monitoring, battlefield surveillance and reconnaissance, or even interactions between autonomous automobiles. As a case study to highlight constraints imposed in field robot applications, particularly in irregular and unstructured environments, I examine the *urban search and rescue* (US&R) domain in Section 1.1.

Many application domains reward the use of small and cheap robots in terms of mission efficacy and deployment feasibility, but the the size, weight and power (SWaP) constraints on these robot platforms constrains robot autonomy. Smaller robots can accomplish some tasks better in some mission scenarios, such as being able to maneuver through smaller openings. Cheaper robots make it not only financially feasible to deploy a larger team to gain coverage benefits, but also allow for greater risk to be taken by individual robots because each individual is nearly disposable. However, these small platforms have limited payload capacity for more capable sensing, computation and communication resources. As is frequently the case in consumer electronics, batteries limit the power available for autonomy, and relies on more passive sensors, slower processors, and shorter range communication.



(a) Centralized. (b) Hierarchical. (c) Decentralized.

Figure 1: Comparison of network architectures over a team of 5 robots. The centralized structure (left) has a common fusion point of robot A and direct communication to all other robots. A hierarchical structure (center) deputizes an additional robot C to act as a fusion center for robots D and E. The decentralized structure connects adjacent robots, reducing the effect of communication or node loss.

This dissertation focuses on a key enabling problem for robot teams: online, robust and cooperative perception. The basic perception problem is the process by which a robot combines information from its typically noisy and imprecise on-board sensor systems into a robust model of the environment. I formulate this perception problem an example of the *simultaneous localization and mapping* (SLAM) problem (as described in surveys [137, 12, 51]), which uses probabilistic techniques to estimate a map of the environment and track the position of a robot. Because robots are using the maps they generate to make decisions, the solution needs to be *online*, such that the robots are maintain a SLAM solution as they move through the environment. In a *cooperative* multi-robot SLAM scenario, I exploit the presence of a team of robots operating in a common environment to improve the quality of the SLAM solution over what an individual robot could accomplish.

The key to enabling robust autonomy in these teams of small and cheap robots is to use collaboration through shared measurements of the environment so that the team of robots can assemble a clearer map of the environment than an individual robot. However, the capabilities, particularly in terms of scalability and robustness, of a multi-robot perception system is dependent on how information is shared between platforms. The *network graph* showing connectivity between individual robots in a

team provides a means to analyze robustness of the approach.

The most straightforward approach to multi-robot SLAM is *centralized* data fusion, in which all robots transmit their sensor data back to a central location, illustrated in Figure 1a, which can then send back an updated global map of the environment to the robots in the team. While this approach will generate the best quality map combining all of the available sensor measurements, it requires the presence of a reliable and fast network connection between all of the robots. However, in environments with low-power network connections, relying on a strictly centralized approach to collaborative perception will not be sufficiently reliable. Variations on centralized approaches exist, such as hierarchical networks, as shown in Figure 1b, which still maintain fusion centers, or the complete network graph, in which all robots communicate with all other robots.

The key to developing robust robot team is *decentralized* communication, illustrated in Figure 1c, between individual robots, where each robot shares information with those robots in its local neighborhood of robots within communication range. Originally formulated by Durrant-Whyte [49, 50] and further developed through work on sensor networks, this approach is known as *decentralized data fusion* (DDF), which provides estimates for individual nodes in a networked system. The key metrics for such a system are survivability under both communication and node failure, while minimizing the per-node computation and communication bandwidth. Employing DDF techniques shifts from creating a centralized, global map for all robots to extending the *sensor horizon* of individual robots with neighborhood map information.

By using a decentralized network architecture, I can relax computational and communication constraints on the system, which improves scalability to larger teams, as well as the robustness to failure. By limiting the interaction of each robot to a subset of neighboring robots, I can bound complexity costs as more robots are added to the team. In a centralized approach, because all messages must first reach

Figure 2: Common hazard US&R cases, taken from the Disaster City training facility in College Park, TX, with a hanging slab over a collapsed car park (left) and a partially collapsed building with flattened floors.

a central location, there exists a single point of failure which makes the system as a whole brittle to communication or node failure.

A key characteristic of the data fusion process is decoupling the problem into a *data association front-end* specific to the sensing modalities of a given robot and an *inference back-end* that fuses observations into a consistent belief model of the environment. The front-end aspects the SLAM problem vary significantly by application domain, but the inference machinery forms a more generalizable problem using abstractions afforded by probabilistic modeling, which will comprise the majority of the work presented.

1.1 Case Study: Urban Search and Rescue

While there are a variety of domains for large scale multi-robot systems, I will use US&R as a canonical example, as it has been studied extensively [107, 104, 105, 106, 123, 122] and illustrates many of the system constraints targeted by this work. The use of teams of small and cheap robots particularly aids in rapid structural and hazard assessment, such as in the aftermath of large scale disasters such as earthquakes or tsunamis. Large teams of robots not only allow for faster coverage of an area, but greater robustness to failure due the redundancy of the robot team itself. In the early stages of approaching an incident site, responders need to assess of buildings on

both the possibility of trapped survivors, as well as hazards to the responders upon entry [8], which typically requires a slow, methodical and dangerous shoring process to ensure safety [9].

In currently in use rescue robotics approaches, human responders deploy a robot and then directly teleoperate the robot through the environment allow the operator to search for survivors and identify hazards. In most cases, the robots are tethered to provide a sufficiently reliable communication link for teleoperation when the robots go out of sight of human spotters. Because of the extreme variability of terrain in disaster sites, navigation and mobility is typically challenging, especially when combined with the limited fields of view from on-board cameras. Failures of different components are common, such losing communications or power due to a broken tether, which frequently result in robots being lost within disaster sites. Any system I design for use in these applications will need to account for the possibility of robot failure, preferably without also causing the whole mission to fail.

In the case of ground robots currently in use, like those used for exploring rubble or defusing explosives, the typical ratio of humans to robots is $2:1$, with an even higher ratio for aerial robots [108] due to the need for spotters. Enabling autonomy in robot teams can help improve this ratio, where a small number of operators can control a large fleet of robots. However, the development of reliable autonomous systems remains a substantial challenge, particularly with robots in dangerous, uncertain environments.

While robot teams provide a means to cover large areas faster, this will require a transition from robots that are directly teleoperated by responders to teams of robots working semi-autonomously. In a semi-autonomous robot team, individual robots can perform basic tasks like navigation and exploration autonomously, and request human intervention to handle challenging cases, thereby minimizing the amount of interaction human operators need. As demonstrated in the Multi Autonomous Ground-robotic

International Challenge (MAGIC) 2010 competition, the winning team deployed a team of 14 semi-autonomous robots supported by only two human operators in a large scale mapping and surveillance exercise [120, 124].

1.2 Research Question and Thesis

This work presents an decentralized approach to inference back-end portion of the multi-robot SLAM problem that is applicable to large robot teams. The core research questions investigate how to develop inference algorithms under the constraints posed by perception for multi-robot autonomy in possibly hazardous environments.

1. How can we approximate a centralized SLAM solution in an online decentralized network, such that each robot has access to a larger map than is locally available?

2. How can we ensure map information is not lost to the rest of the network in the event of a communication or node failure?

3. How can we ensure that map estimates on robot nodes are not over-confident due to double-counted information?

4. How can we bound the costs of computation and communication per robot as the size of a robot team increases?

In addressing these research questions, I developed an overall approach for marrying the single-robot inference technique of *smoothing and mapping* (SAM) [42] with networked inference techniques to create a solution for the DDF problem. The result is DDF-SAM, in which robots choose relevant variables to share with neighboring robots as summarized maps, which can be transmitted and cached across the network as robot fuse them with locally available measurements. By fusing these maps from its neighbors, each robot can extend its sensor horizon, which make the map representation more useful for exploration and navigation tasks.

The thesis of this dissertation is as follows:

DDF-SAM provides a smoothing and mapping solution to inference in the decentralized data fusion problem that is online, scalable and consistent, while flexible enough to provide an extended sensor horizon under a variety of sensing modalities.

I can break down the core requirements for an effective decentralized inference technique as follows:

Online An online map solution entails maintaining an estimate during runtime, as opposed to offline techniques which collect all available data and process at a later time. An online map estimate is necessary to enable basic autonomy, such as navigation.

Scalable In order to allow for large teams of robots with wide coverage, the inference algorithm needs to be scalable in terms of how per-robot communication and computation costs increase with each additional robot to the team.

Consistent A requirement for the correctness of any given estimate is it does not become overconfident due to double-counted measurement information, which can impose constraints on handling of measurements or on network architecture.

As a scope limitation to this work, I will not integrate data association into DDF-SAM, so as to focus on the decentralized inference problem. I seek to ensure that the approach is flexible enough to accommodate a variety of sensor types, computational facilities, and network capabilities. A large part of this flexibility comes from decoupling the data association process from the inference back-end, where the same inference technique works for different front-end approaches.

In order to demonstrate the usefulness of the system, under the scope limitation that the work presented addresses pure perception problem without closing a control loop in a robot system, I use extended sensor horizon as a proxy for measuring the

usefulness of the approach. This is reasonable because a more complete map with allow exploration or navigation algorithm to make more informed choices.

1.3 Contributions

The core contributions of the work in this document are as follows, with corresponding chapters.

- I introduced the DDF-SAM paradigm for decentralizing multi-robot inference by creating and sharing summarized maps based on a chosen subset of locally observed variables. Chapter 3 presents the approach and details the general theory behind DDF-SAM.

- I developed DDF-SAM 1.0, which uses constrained optimization to fuse summarized maps into a consistent neighborhood map. Chapter 4 applies this approach as a batch smoothing process as DDF-SAM 1.0, with evaluations using both simulated and real-world data.

- I developed DDF-SAM 2.0 as a reformulation of map fusion allowing for combination with incremental SAM (iSAM) [86, 85] techniques and enforced map consistency with explicit downdating. Chapter 5 presents DDF-SAM 2.0 in a linear scenario with experimental results.

- I extended DDF-SAM 2.0 for full nonlinear systems using relinearization of previously linearized factors that allow for unknown global positions of robots. Chapter 6 presents nonlinear DDF-SAM 2.0 with experimental results.

- I developed and evaluated several approaches for exact and conservative approximate map summarization techniques. Chapter 7 describes the approaches evaluated, with experimental results.

Before introducing DDF-SAM, Chapter 2 presents background material on robot perception, including the basis for smoothing and mapping algorithms, as well as

multi-robot mapping techniques and requirements. As a further analysis of the results and implications on how to apply DDF-SAM in real-world multi-robot scenarios, Chapter 8 provides a discussion of design trade-offs. Chapter 9 provides an overview of and discussion of the approach, as well as future work and possible extensions.

Chapter 2

PROBLEM BACKGROUND

This chapter provides a general background on robot perception and the domain of multi-robot collaborative perception techniques that form the basis of the work presented in this dissertation. Because a multi-robot perception system is inherently a large system comprised on single-robot perception systems, this chapter starts with standard SLAM techniques, including geometric and probabilistic representations and inference algorithms.

2.1 Simultaneous Localization and Mapping

In autonomous mobile robotics, the core needs of planning and control approaches can be described in terms of *localization*, in which a robot platform estimates its position and movement in a known environment, and *mapping*, in which a robot estimates a map of its surroundings while undertaking known movement. However, in many scenarios, neither the structure of the environment or the robot movement is fully known, leading to the *Simultaneous Localization and Mapping* (SLAM) problem. In recent years, the SLAM problem has had a flurry of work, with surveys available [137, 12, 51], with many techniques and variations, which this section will address.

While the SLAM problem has a variety of formulations, depending on the application – even not specifically within robotics – there is significant common structure in the underlying probabilistic inference problem within each of these domains. This theoretical underpinning connects these techniques for SLAM to other fields, such as scene reconstruction in computer vision. By using the abstraction of a inference over a sparse graphical model, I can easily generalize to different domains and physical sensing modalities.

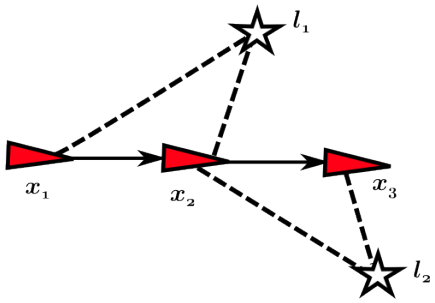


Figure 3: Single-robot landmark SLAM scenario, where triangles indicate robot poses, stars represent landmarks, and lines indicate measurements.

2.1.1 Geometric Representation

The classic model of SLAM is *landmark SLAM*, which estimates the trajectory of the robot as well as the positions of landmarks in the environment. Figure 3 illustrates a simple case with three robot poses, and two landmarks, with corresponding observations and movement. Constraints in landmark SLAM come from measured robot movement (through wheel odometry, control inputs, known motion models or inertial sensing), and observations of landmarks via extracted features. In this SLAM formulation, the primary variables of interest are individual *poses* $x_i \in X$, which are discrete-time samples of the robot trajectory, and *landmarks* $l_j \in L$, which are discrete structural features in the environment that can be observed by a robot. In general, robot poses are oriented points in space, such that there is a translation and and rotation component. In a typical planar environment $X \triangleq \mathcal{SE}(2)$ and $L \triangleq \mathbb{R}^2$, where poses have 3 degrees of freedom (DoF) and landmarks have 2. In the more general non-planar case, $X \triangleq \mathcal{SE}(3)$ and $L \triangleq \mathbb{R}^3$, in which each pose has 6 DoF and each landmark has 3 DoF. To denote variables associated with separate robots, I use superscript notation, such that the trajectory of robot r is $X^r \triangleq \{x_i^r\}$. For convenience, let the set of all variables be defined as $\Theta \triangleq \{X, L\}$.

The choice of geometric representations depends on the application, with simpler indoor ground robot scenarios using planar representations and outdoor or flying robots using general poses. Reducing the number of degrees of freedom allows for a

more compact representation that is computationally cheaper to use for SLAM. Furthermore, depending on the application, landmarks are generalizable to any geometric representation of the environment suitable for both the application and sensing capabilities of the robot, such as extracted image features corresponding to fixed points in space or even higher order features such as line segments, planes or objects [59, 141].

One key representational challenge appears when computing the effect of a small change $\delta \in \mathbb{R}^n$ to an existing variable $\theta \in M$, where $M$is a manifold entity such as a rotation or pose, which will generalize through the use of manifold retractions. As an example, consider the case of the of a one-dimensional rotation θ in radians, where 0, 2π and 4π correspond to the same final rotation. When applying an update δ to θ, if using simple addition, it would be necessary to include an extra step to bound the rotation back to $-\pi \leq \theta \leq \pi$, which induces an additional nonlinear calculation. Instead, I use the manifold representation $\theta \in \mathcal{SO}(2)$, where a retraction operation $\oplus : M \times \mathbb{R}^n \to M$ exists to apply a change $\delta \in \mathbb{R}^n$ to an existing θ_0 to yield another $\theta_1 \in M$. In this case, the space $\mathbb{R}^n$ forms the tangent space around an existing θ_0, and provides a smooth mapping between θ_0 and θ_1, where n is the degrees of freedom of the manifold M. For a complete formal treatment of manifold operations, see [48, 47]. For the more specific applications of geometric manifolds to probabilistic inference and general optimization, see [1].

There are a few other manifold operations useful for representing geometry, as well. Let T_A^B denote a transform mapping a variable θ_i^A in the reference frame of A to a corresponding version θ_i^B in the reference frame of B. I can compute this transform as a variable, where for planar cases, $T_A^B \in \mathcal{SE}(2)$, and applying this transform is an operation $T_A^B\left(\theta^A\right) = \theta^B$, and depending on the type of θ, is a Lie group compose operation for poses $T_A^B\left(x^A\right) = T_A^B \circ x^A$. As an abuse of notation, let $\mathcal{I}$ denote an identity transform. As notational conveniences, I define a *between* operation between manifold objects from manifold M as $\ominus : M \times M \to M$ such that

$\theta_a \ominus \theta_b = \theta_b \circ (\theta_a)^{-1}$. As an intuition for this operation, consider $M = \mathbb{R}$, where a compose operation $\theta_a \circ \theta_b \triangleq \theta_a + \theta_b$ and an inverse operation is simple negation, the between operation is $\theta_a \ominus \theta_b \triangleq \theta_b - \theta_a$.

2.1.2 Example SLAM Domains

This section provides some examples of how SLAM can be applied across different problem domains and sensing modalities, as well as highlighting connections with other research areas. The important conclusion from examining these approaches is that same basic graphical abstractions apply in each of these cases.

2.1.2.1 Visual SLAM

The clearest example of landmark SLAM is the visual domain, particularly exemplified through the work of Davison, et al. with real-time monocular SLAM [37], where rather than using the entire contents of an image for reconstruction, feature extraction finds a set of 2D points on the image to act as representative samples. There is a wide body of work in computer vision literature feature extraction from images, such as Harris corner features [66, 145], "good features to track" [130], SIFT [94], and SURF [16]. In the case of visual features, a frequent design goal is to find image features that correspond to geometric features in the environment, such as corners and edges. Another route to finding representative structural features in the environment is to augment the environment (or other robots) with markers designed to facilitate use as a measurement in a SLAM system, as well as ensuring landmarks are uniquely identifiable. The most commonly used approach is planar fiducial markers [54, 143, 118].

Another variation on a SLAM approach comes from the computer vision community, in the form of the *Structure from Motion* (SfM) problem [67], in which an algorithm reconstructs the geometric structure of an environment from an unordered

collection of images. Notable examples of this domain use large collections of images scraped from Internet databases to reconstruct a relatively dense model of the environment [132], and can be expanded to large scale environments, such as reconstructing the city of Rome [5].

This problem corresponds to a variation of the landmark SLAM problem, solving for the positions of cameras (poses in the landmark SLAM formulation) and landmarks in the environment, but without measured constraints between the cameras.

2.1.2.2 Pose SLAM

One of the more popular approaches to SLAM is the use of scan-matching between to induce a series of constraints operating directly on the poses in a graph, originally deriving from the use of planar laser scan results [96, 95, 26, 27]. In this case, the "map" can be generated implicitly given the poses by back-projecting scans into the environment. This approach has been used extensively [116, 117], particularly with the recent introduction of affordable 3D laser scanning sensors, which has made 3D pose SLAM approaches viable [115, 114].

This approach has a similar structure to the landmark SLAM problem, in which the only variables of interest are the poses of the robot in the environment, and constraints derive from robot motion or from relative displacements between spatially close poses computed via scan matching.

2.1.3 SLAM System Architecture

The basic structure of a single-robot SLAM system has two phases:

1. *Data Association Front-End*: This initial phase finds correspondences between newly observed features and previously observed features. These correspondences can come in the form of labellings for discrete features, as well as relative pose constraints that match a current sensor scan against a previous scans.

2. *Inference Back-End:* This phase updates the current belief for variables in the system by fusing information from measurements with prior information.

While the SLAM community typically treats this phases as separate system components, though there are techniques to both data association and inference at the same time. These techniques treat correspondences as explicit random variables to estimate, leading to a mixed discrete/continuous problem, which is well-solved by Expectation-Maximization (EM) techniques [140, 43]. Monte-Carlo sampling techniques provide another robust means to determine feature correspondences as a part of an estimation problem [38, 44, 39]. A classical technique is to use robust statistical error models [71] rather than the more standard Gaussian models for measurement error to mitigate the effect of having spurious correspondences. This approach of making the inference problem more resilient to outliers measurement correspondences has received attention recently, incorporating both robust and multi-modal measurement models [119, 2, 126].

As a core assumption for he majority of this work, however, I focus on the case of either known feature correspondences, or a decoupled system, such that the inference back-end assumes all correspondences are correct. The reason for this decoupling assumption is to simplify the analysis of the inference portion of a system, which will become more important when evaluating a multi-robot system due to the added layer of complexity added by a multi-robot system.

2.2 Data Association

The data association problem, also known as the correspondence problem, consists of matching new measurements of a feature to either an existing landmark in the map, or initializing a new landmark. Many techniques based on features extracted from sensor inputs rely on feature descriptors, typically computed during feature extraction, and use machine learning techniques to cluster and compare newly extracted features

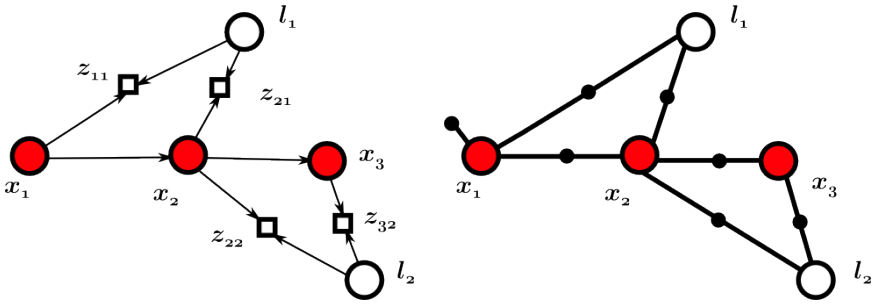
with previously observed features. Because the proposed work focuses on the more generalizable back-end inference problem of SLAM, the remaining work presented assumes either labeled features (like fiducials) or completely unlabeled features.

The single robot data association has been studied extensively, with techniques ranging from Nearest Neighbor [12] and maximum likelihood [103] to more sophisticated combinatorial optimization approaches, such as Joint Compatibility Branch and Bound (JCBB) [109]. The challenge with the data association problem is that it is a combinatorial optimization problem, which scales poorly as the number of features increases. Some approaches cast the data association problem as an assignment problem and use the Hungarian algorithm [55].

Another technique for choosing correspondences is sampling, in which sets of putative correspondences are sampled and evaluated, returning the set of correspondences with the highest score. This approach forms the basis of Random Sample Consensus (RANSAC) [56] and variations, which has become a staple algorithm in computer vision for efficiently matching features across frames in the presence of outliers, with variations to exploit domain structure, such as groupings [112] or non-holonomic motion constraints [129].

2.3 Probabilistic Inference

The back-end inference problem consists of an inference problem over measurements collected as the robot drives through the environment, in which the goal is to maintain a probability distribution over the variables of interest (poses X, landmarks L), while incorporating information from measurements $z_i \in Z$ that have been corrupted by noise. The general form of this problem can be expressed as a conditional probability density on the posterior $p(X, L|Z)$, which can be difficult to maintain and represent directly. To simplify the notation, let $\Theta \triangleq \{X, L\}$, such that the distribution of interest is $p(\Theta|Z)$.



(a) Bayes Network (b) Factor Graph

Figure 4: Graphical model representations of simple SLAM example in Figure. 3, with a directed Bayes Network model (left) and the equivalent undirected bipartite Factor Graph model.

The Bayesian approach to this problem is to use Bayes law to invert the conditional to form a better-constrained problem,

$$\begin{aligned} p(\Theta|Z) &= \frac{p(\Theta)\,p(Z|\Theta)}{p(Z)} \\ &\propto p(\Theta)\,p(Z|\Theta) \end{aligned}$$

which implies the combination of a prior $p(\Theta)$ on the variables of interest, and a generative model in the conditional $p(Z|\Theta)$, in which one can predicted sensor values given the state of the world, and evaluate the likelihood of a given configuration of poses and landmarks by comparing the predicted measurements with the actual measurements.

The Bayesian approach to inference uses Bayes law to invert the conditional to yield a density amenable to factorization by measurement modality, such that the key density is $p(\Theta)\,p(Z|\Theta)$, combining prior information $p(\Theta)$ with conditionally independent generative measurement models $p(Z|\Theta)$. The generative model approach allows for the use of measurement models derived from each measurement modality available to the robot, such that for each sensor, there is a function $h(\Theta) = Z$ that can predict the sensor output given state variables.

Figure 4 illustrates this representation, in which individual measurements z_i have corresponding prediction functions $h(\Theta_i) = z_i$, such as the landmark observation

measurement z_{21} being generated (as denoted by arrows) pose x_2 and l_1, leading to the measurement density $p(z_{21}|x_2, l_1)$. Robot motion is frequently represented as a generative model where the pose at time time t predicts the pose at time $t+1$, written as a probability density as $p(x_{t+1}|x_t)$, and illustrated in a Bayes network with the arrows from one pose to the next. The overall inference problem of incorporating new, noisy measurements into an estimation of $p(\Theta|Z)$ is the key problem underlying the SLAM techniques presented. The key variations in techniques come from the representation of the probability density, assumptions on how noise affects the system, and the factorization of the probability density.

The following sections summarize the standard techniques for probabilistic inference as used in SLAM applications, starting with the more classical filtering approaches in Section 2.4, and then moving on to smoothing approaches used by the rest of this work in Section 2.5.

2.4 Filtering Approaches

The classical approach to estimation is the Kalman filter [87] and extensions. The standard Kalman filter represents the system as a multi-variate Gaussian distribution, with a mean μ and covariance matrix Σ, under the assumptions of a linear measurement model and measurements corrupted with zero-mean additive Gaussian noise. Because the measurement and prediction functions are assumed linear and the uncertainty is assumed Gaussian, the inference operation can be performed with linear algebra operations to yield an optimal estimate.

In more realistic scenarios for robot mapping, in which measurement functions are nonlinear, the *Extended Kalman Filter* (EKF) adds a linearization step, which uses a linear approximation to perform updates. The classic SLAM solution [131] uses an EKF to estimate the robot pose and the positions of landmarks, which has

been the basis for a great deal of further work. Managing nonlinearity in this filtering framework has been a central challenge, and approaches have included finding more accurate linearizations through the unscented transform to form the *Unscented Kalman Filter* (UKF) [78, 77], as well as reparametrizing the state to minimize the effect of linearization error [133].

In addition to the errors induced as a result of measurement nonlinearities, Kalman filters have difficulty scaling due to the need to maintain a large, dense covariance matrix which grows in size as as the robot explores. The key insight to addressing this problem is to exploit the *sparsity* of the problem to perform more efficient calculations. This problem motivates different parameterizations of the underlying linear density, such as using the information form of a Gaussian density, either maintaining an information matrix $\Lambda \triangleq \Sigma^{-1}$ and information vector $\xi \triangleq \Sigma^{-1}\mu$. Information-space formulations have the advantage of additive updates, and in the square root form, they maintain sparsity. The Sparse Extended Information Filter (SEIF) [17] is a filtering approach that combines the information filter with active sparsification step to maintain the sparsity of the system [53, 52, 146] and has been proven to be efficient enough for real-time use in a variety of SLAM scenarios, in both single-robot [139] and multi-robot cases [138].

Another approach to manage nonlinearity is nonparametric multi-hypothesis techniques, such as particle filters, which maintain a large number of weighted possible solutions. Many approaches, most notably FastSLAM [102, 103], combine particle approaches with EKFs, such that each particle maintains its own EKF solution, while facilitating maximum likelihood data association for real-time use [15].

2.5 Smoothing and Mapping

Originally conceived as an offline approach, graphical SLAM techniques operating directly on constraint graphs have gained traction due to advances in sparse linear algebra techniques. In contrast to filtering approaches described in the previous section, graphical SLAM techniques explicitly maintain the density in terms of both the variables Θ and measurements Z. This paradigm has enabled the SLAM community to apply techniques from sparse linear algebra and graph theory to improve performance in batch solving cases or allow for efficient incremental solutions. This graphical approach has produced a large variety of work, including batch [40, 42, 116, 64, 63, 41, 137, 58] and incremental solvers [83, 84, 86].

The following sections describe the core representations of probabilistic inference for SLAM, introducing factor graphs as a sparse factorization of measurements, the variable elimination algorithm for performing inference and marginalization, and the Bayes tree as a partially solved form of a factor graph after elimination. For a more detailed treatment, see [42, 83, 85].

2.5.1 Graphical Inference as Optimization

This representation the probabilistic inference problem converts the underlying Bayes network, as in Figure 4a to an undirected graph called a *factor graph* [91], shown in Figure 4b. In this model, a *factor* $\phi(\Theta_i; z_i)$ is a non-negative, real-valued loss function operating on a subset of variables $\Theta_i \subseteq \Theta$, corresponding to error between the predicted measurement and a real measurement. Combining many of these factors forms the full factor graph $\Phi(\Theta; Z) \triangleq \{\phi(\Theta_i; z_i)\}$. This representation enables reformulation the in negative log-likelihood form to generate an additive loss function for the full system

$$L(\Theta) \triangleq \sum_i \phi_i(\Theta_i; z_i) \propto -\log \prod_i p(\Theta_i | z_i).$$

Casting the inference problem as a large-scale least-squares optimization problem, and in terms of a prediction function and under the assumption of measurements corrupted by zero-mean, Gaussian noise with covariance Σ, yields the full optimization problem in graphical form:

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \ \frac{1}{2} \sum_i \left\| h\left(\Theta_i\right) - z_i \right\|_\Sigma^2 \tag{1}$$

In general, the measurement function $h\left(\Theta_i\right)$ will be nonlinear, which is typically solved through a process of successive linearizations of the problem to compute updates δ^k to an initial linearization point Θ^k, such that each iteration is $\Theta^{k+1} = \Theta^k + \delta^k$. More sophisticated nonlinear optimization techniques, such as Levenberg-Marquardt, apply a form of damping to the linear system to prevent overshooting the solution in highly nonlinear domains. The linearized form of a factor graph forms a large sparse system as follows, in which H is the Jacobian $h\left(\Theta_i\right)$ evaluated at Θ_i^k,

$$\frac{1}{2} \sum_i \left\| H\delta^k + h\left(\Theta_i^k\right) - z_i \right\|_\Sigma^2 \tag{2}$$

$$\frac{1}{2} \left\| A\delta^k - b \right\|_\Sigma^2 \tag{3}$$

where the block-sparse matrices A and b are the *square-root information* form of the system, in which $A^T A$ is the information matrix. This form of the system, in which factors can be combined and manipulated with linear algebra operations, allows for several operations that form the basis for techniques presented in preliminary and proposed research.

The resulting linear system can be written as a single block-sparse linear system, as in (3), which maintains the same sparsity structure as the underlying factor graph. In a batch formulation, this sparse linear system can be solved using standard sparse linear algebra techniques, such as QR or Cholesky factorizations. However, the details of solving this linear system facilitate key operations enabling incremental solving and efficient marginalization, which the following sections will describe in greater detail.

This SAM framework for probabilistic inference using factor graphs provides a basis for a variety of techniques, and will correspondingly underly the proposed multi-robot SLAM work. Note that through incremental solving techniques, it is possible to gain the benefits of filtering approaches while maintaining a more explicit representation of the measurement structure.

2.5.2 Variable Elimination

The *elimination* algorithm is a way to factorize a factor graph of the form into a Bayes net of the form

$$p(\Theta) = p(\theta_1|S_1)\, p(\theta_2|S_2) \ldots p(\theta_n) \tag{4}$$

where S_j denotes an assignment to the *separator* $\mathcal{S}(\theta_j)$ of variable θ_j, defined as the set of variables on which θ_j is conditioned, after elimination. The elimination algorithm, closely following the notation in [89], although using a different narrative where the end-result is a Bayes net. Let $\Phi_{j:n} \triangleq \phi(\theta_j, \ldots, \theta_n)$ denote a partially eliminated factor graph. The algorithm proceeds by eliminating one variable θ_j at a time, starting with the complete factor graph $\Phi_{1:n}$. Eliminating a single variable θ_j yields a single conditional $p(\theta_j|S_j)$, as well as a reduced factor graph $\Phi_{j+1:n}$ on the remaining variables. After all variables have been eliminated, the result is a Bayes net with the factorization in (4). The computational cost of this algorithm depends strongly on the variable elimination order, in which orderings yielding small separator set sizes reduce the cost of each elimination operation.

Partial elimination, which yields a factor graph $\Phi_{j:n}$ on a smaller set of variables, serves as a means to create a joint density over a specific set of variables. To create a factor graph on a set of variables Θ_A, choose an ordering such that $\Theta \setminus \Theta_A$ appear first.

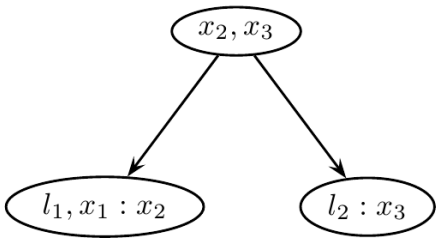


Figure 5: Small example Bayes tree, showing a root clique with two variables, and two leaf cliques.

2.5.3 Incremental Solving with the Bayes Tree

With a linear system factorized through elimination, I exploit the structure of the resulting Bayes net to allow for incremental solving [86] of a *single solver*, in which I update only sections of an existing Bayes net as I add new factors to the system.

The structure of the Bayes net in an incremental solver has as a more general structure: the Bayes tree [83, 85] is a tree-structured graphical model that defines a factored density over cliques of variables. Each node in the tree defines a conditional density conditioned on its parent, in the much same way as a Bayes net. However, unlike Bayes nets, a Bayes tree is always acyclic, and its nodes can have variables in common. The formal definition of the Bayes tree is given below. Given a set of variables Θ, a *Bayes tree* $\mathcal{B} \triangleq (\mathcal{C}, \mathcal{E})$ is a rooted tree whose nodes represents *cliques* $C_i \subset \Theta$, while its structure defines the following joint density $p(\Theta)$ on the variables Θ:

$$p(\Theta) = \prod_i p(F_i|S_i) \tag{5}$$

Here the *separator* S_i is defined as the intersection $C_i \cap \Pi_i$ of a clique C_i and its parent clique Π_i, the *frontal variables* F_i are the remaining variables in C_i, i.e., $F_i \triangleq C_i \setminus S_i$, and $p(F_i|S_i)$ is a conditional density on F_i given S_i. An example is shown in Figure 5, where I write $C_i = F_i : S_i$ for a clique C_i.

2.5.4 Computing Marginals

The key is using the recursive structure of the Bayes tree to compute only the marginals $p(S_i)$ on the separator sets S_i, via

$$p\left(S_i\right) = \int_{\neg S_i} p\left(F_\pi | S_\pi\right) p\left(S_\pi\right) \tag{6}$$

where I informally used $\Pi_i = F_\pi : S_\pi$. To avoid redundant computation, I cache the separator marginals $p(S_\pi)$ so each is only computed once. I can then compute the clique marginals $p(C_i)$ by $p(C_i) = p(F_i|S_i)p(S_i)$.

Fine-grained marginals on an individual variable θ_j can always be computed by $p(\theta_j) = \int_{\neg\theta_j} p(F_i)$ for the (unique) clique C_i where θ_j is a frontal variable $\theta_j \in F_i$, and where the frontal marginals $p(F_i)$ can be computed using with clique or separator marginals:

$$p(F_i) = \int_{S_i} p(C_i) = p(F_i|S_i) \int_{S_i} p(S_i)$$

If the marginal for more than one variable in F_i is required, it is beneficial to cache the calculation of $p(F_i)$.

2.6 Networked Inference

When moving to multi-robot systems, it is common to directly extend single-robot SLAM techniques, though the addition of multiple robots and limited communication introduce additional challenges increasing the level of complexity in analysis. In the *online* mapping scenario, in which the robots perform SLAM while exploring the environment, rather than in post-processing, the distinguishing characteristic of multi-robot system is *network topology*, which determines how robots share information. See [101] for analysis of distributed systems with graph theoretic techniques.

Within multi-robot and sensor network domains, there are a wide variety of communication modalities, and I would like to extract key characteristics from these

techniques to model with robot network abstractions. Common types of robot communication include radio-frequency wireless communication devices, such as high-bandwidth consumer-grade IEEE 802.11 or low-power short-range IEEE 802.15 devices (including Zigbee and Bluetooth consumer versions), acoustic modems for underwater applications. These technologies vary in data rate, effective communication range, robustness to varying conditions. For the purposes of this work, I assume bidirectional communication unless otherwise specified.

To characterize the robot network, I use the concept of a *local neighborhood* of a robot, which depends on the networking capabilities and configuration of robots. Let N_r denote the set of robots that communicate with r, such that $r \notin N_r$, and let N'_r be an augmented neighborhood $N'_r \triangleq \{N_r, r\}$. In practical scenarios, this is a time-varying set due to communication and node failures, which results in a dynamic network topology. The robot network can then be modeled as an undirected graph $\mathcal{G} \triangleq \{\mathcal{R}, \mathcal{E}\}$, where nodes are the set of robots $\mathcal{R}$ and edges are the set of bidirectional communication links $\mathcal{E}$. In the case of time-varying graphs $\mathcal{G}(t) \triangleq \{\mathcal{R}, \mathcal{E}(t)\}$, it is the edge set that varies over time as individual robots move in and out of communication range.

The remainder of this section provides background on inference algorithms designed to work with the robot networks, starting with centralized inference approaches in Section 2.6.1 as a baseline solution. Section 2.6.2 highlights a key challenge for moving to decentralized approaches, which is the information double-counting problem. Section 2.6.3 reviews prior work on decentralized inference techniques.

2.6.1 Centralized Inference Approaches

A natural extension of single-robot techniques to multi-robot SLAM is to have robots send measurements to a fusion center to build a single map containing measurements from separate robots. With sufficiently reliable and fast communication on each robot,

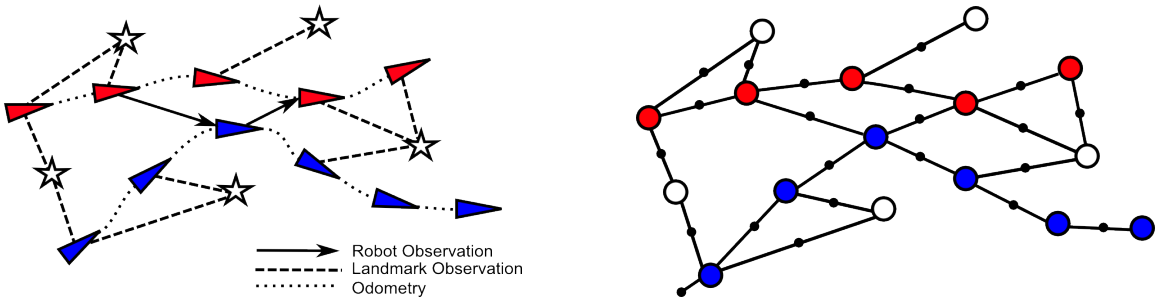


Figure 6: Simple multi-robot SLAM scenario, with two robots and common landmarks, and a corresponding factor graph (right). In this naive factor graph representation, poses are colored circles, landmarks are white circles, and all measurements are connected with factors.

this is demonstrably feasible [120, 124], especially if the central location has access to greater communication resources. Centralized graph-SLAM techniques allow for easier extension of single-robot SLAM algorithms, such as iSAM [86] applied to multi-robot mapping via pose-graphs [88], or C-SAM [6] which combines observations from multiple robots in a single factor graph. Figure 6 illustrates a multiple robot scenario converted into a centralized factor graph. Other approaches include particle filters [24, 60] and manifold representations [68].

Another variation on centralized approaches derives from the fusion of using multiple local maps, which has received a lot of traction in a single-robot context [30, 136, 18, 25], as it leads to computationally more efficient algorithms. In addition, as mentioned by Tardós et al. [136], local maps lend themselves naturally to multi-robot mapping, as strategies for map-merging can just as well serve to merge maps built by different robots.Tectonic-SAM [113] uses a divide and conquer approach with locally optimized submaps, recursively partitioned by choosing an elimination ordering using nested dissection.

I define a centralized system based on the existence of a *fusion center* r_c appearing in the network (either as an mobile platform or an operator station), such that the $N'_{r_c} = \mathcal{R}$, in which data fusion success is dependent on maintaining communication between r_c and N_{r_c}in order for r_c to have a full global map estimate. Note that it

is possible for a system to be both distributed and centralized, as in the case when individual nodes in a system perform local inference and then transmit local solutions back to an operator station for global map fusion, as in [69]. The significant detail is the presence of and reliance on a node having a full, global map. A dual form of the centralized network is when $\mathcal{G}$ is complete, that is, all robots share information with all other robots simultaneously. In this case, each robot r meets the requirements for a fusion center node.

Using centralized approaches introduces key problems in system resilience and in scalability to large teams of robots. The fusion center in a system can become a single point of failure for the robot network, as if this node either becomes disconnected from the rest of the network or is disabled, the rest of the robot team cannot function. Furthermore, having a fusion center can become a bottleneck for system performance due to network congestion. Because the fusion center maintains a full global map as a part of its inference problem, online operation can become bound by its ability to update the global map.

I analyze scalability by examining performance costs for each new robot added to the team. In a centralized configuration with a single fusion center r_c, adding a new robot linearly increases the number of communication links to maintain for r_c (either via direction connection or multi-hop message passing), as well as increasing the size of global map estimate. In the complete network graph configuration, this increase is mirrored by all platforms.

2.6.2 Information Double-counting

One of the implicit requirements for data fusion is that separate measurements from a sensor need to be statistically independent. Fusing multiple measurements that are correlated results in the *information double-counting problem*, in which an estimate can become overconfident, which in the case of maps used for robot navigation, could

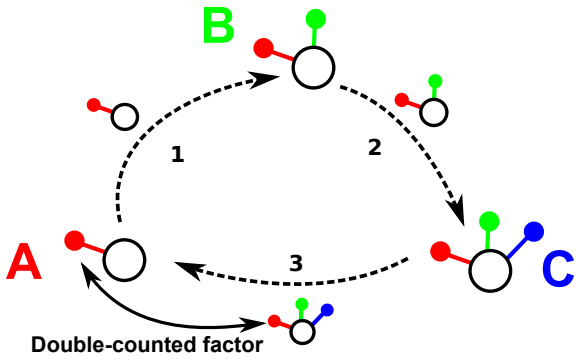


Figure 7: Double-counting illustration with three robots, showing cyclic message passing. Information, denoted as factors from each robot, accumulates in the message passed between robots, leading to robot A adding a duplicate copy of its contributed factor if the message in step 3 is added naively to the current estimate of A.

result in a robot making erroneous navigation decisions. In single robot SLAM, correlated measurements can result in incorrect or overconfident estimates, such as in scenarios where many measurements are recorded while a robot is stationary [81], but in a decentralized network there is a much greater chance for double-counting to impact estimates.

In order to expand to a scenario in which robots share fused information, rather than raw measurements, it is necessary to examine the independence assumptions between measurements. The primary challenge in ensuring consistency is avoiding overconfidence due to *double-counting* measurements as information propagates through the network.

Consider the following example with three robots A, B, and C, illustrated in Figure 7 with simple message passing in which robots add factors to the shared estimate. By completing this cycle in the network, robot A has received a new estimate incorporating a copy of its local information. If robot A treats the message from robot C as independent of its own measurements, it will double-count its local measurements, leading to overconfidence in its estimate.

It is possible to avoid this problem by enforcing non-cyclic constraints on network topology [147], i.e., preventing the message in step 3, but this reduces the resilience of

the system to communication and node failures. Other work has focused on managing the bookkeeping necessary to prevent double counting information [28] in filtering contexts, while relying on managing network topology actively.

Covariance intersection [80] is another method for consistent information fusion that explicitly accounts for the possibility of correlated measurements by estimating an upper bound on the uncertainty of the fused information that is provably conservative. For a geometric intuition of this operation, consider the case of fusing two Gaussian random variables A and B: for any amount of cross-correlation between the variables, the resulting covariance ellipse will be contained within the convex intersection of the covariances ellipses of A and B. By using the intersection of the covariance ellipses as an upper bound on the uncertainty of the fused variable, it is possible to fit a conservative covariance ellipse for the fused result by ensuring that the entire intersection of the covariances is contained. This approach has been successfully applied to single-robot SLAM [79]. It is important to note that covariance intersection ensures consistency at the cost of a conservative approximation of the correct density.

2.6.3 Decentralized Inference Approaches

The more robust approach to multi-robot SLAM, particularly when scaling to large numbers of possibly unreliable robots, uses a *decentralized* approach to robustly propagate information and distribute computation between platforms. Note that I want to distribute the computation to minimize the amount of redundant computational work necessary and keep the local inference problem on each robot tractable. This problem is known in general case as the Decentralized Data Fusion (DDF) problem, introduced as a solution for distributed sensor networks [49, 100, 98], which applies additional constraints to the SLAM problem. This general problem places requirements on any solution, such that a DDF system must be scalable under both

communication bandwidth and computational limitations on robot platforms, and robust to robot and communication failure. The robot soccer domain has provided a useful example domain for distributed tracking [135] using probabilistic techniques, which can be extended to allow for planning for information gain [134].

Covariance intersection has been a classic approach for ensuring consistent data fusion across platforms, as it directly handles the possibility of double-counted information from messages between robots. For a an approach using general covariance intersection, see [82]. More recent work expanded conservative fusion techniques to allow for non-Gaussian densities through such techniques as exponential mixture models [76] and the geometric mean density [13].

One commonly addressed subproblem in decentralized inference in robotics is "cooperative localization," originally presented by Roumeliotis and Bekey [127], which solves for robot positions in a decentralized manner using direct observations between platforms. While not a full SLAM solution, these approaches can form a basis for future work. The original formulation [127] uses a single distributed Kalman filter which estimates a pose from all members in a team using available positioning information. Bahr, et al. [10] introduced a technique using multiple AUVs performing mobile trilateration in which they instantiate up to 2^n filters for each of n vehicles to keep track of the sources of vehicle information to prevent double-counting. Nerurkar, et al. [110] presented a distributed MAP estimator using a distributed data-allocation scheme enabling robots to simultaneously process and update local data when equipped with bidirectional sensing of other robots.

Indelman et al. [74] used a graph-based approach to explicitly calculate the required correlation terms for consistent information fusion in the EKF framework while considering a general measurement model. Bailey et al. [11] also uses a graphical approach for cooperative localization, in which individual platforms compute summarized versions of local trajectories and transmit these pre-computed solutions to a

fusion center for combination. The authors note that it is possible to fully decentralize this approach by placing a fusion center on each robot.

Several authors have addressed the full multi-robot SLAM problem and proposed true multi-map, multi-robot algorithms that have several appealing properties [148, 125, 70, 121]. Because minimizing the communication load between robots is important so as to avoid the performance bottleneck of data transfer and to avoid redundant communication, there has been work done to reduce data transfer [111] by choosing the most informative features to transmit.

As is common throughout much of the mapping and estimation community, many techniques employ filtering approaches to perform inference. Roumeliotis et al. [128] consider an extended Kalman filter (EKF) framework and perform a decentralized calculation of the augmented covariance matrix between all the robots in the group assuming relative pose measurements. Thrun et. al, presented an extension of the sparse extended information filter for multi-robot scenarios [139, 137], which actively removed information to ensure sparseness at the cost of approximation. Bryson and Sukkarieh [19] also employ a filtering-based approach for active decentralized SLAM of airborne vehicles.

2.6.3.1 Consensus Approaches

Another approach to decentralized estimation is the class of *consensus* algorithms, which comes from the control community, which acts as a decentralized averaging process. These algorithms, described an analyzed at length by Mesbahi and Egerstedt [101], solve the rendezvous problem for multi-agent control, in which a distributed network of robots converges on a common location, with provable convergence properties under both static and dynamic network graphs. These algorithms also apply to estimation problems, such as in distributed sensor networks, in which the estimate

is the state driven to a common solution. Aragues et al. [7] uses consensus in conjunction with an information filter to merge landmark maps. The consensus solution can be characterized being similar to a distributed preconditioned gradient descent approach (applicable to single-robot SAM cases as described in [75]), which weights the contribution of individual robots to improve convergence.

2.6.3.2 Loopy Belief Propagation

Djugash and Singh presented in [45] an approach in the sensor network domain for decentralized SLAM using only range measurements using a combination of a multi-hypothesis filter and tree-structured loopy belief propagation (LBP). Their earlier work had explored the use of range-only measurements between active nodes, in which some nodes are fixed and some are on mobile platforms, focusing on the multi-modal inference problem [46], but the decentralized inference portion this earlier approach did not account for double-counted information passed through the system. Their solution from [45] to the convergence and double-counting problems using LBP is to choose a tree subgraph from the network graph and perform BP with the standard non-loopy two-pass approach. They use a distributed minimum diameter spanning tree algorithm [20] to choose a subgraph of the current network graph on which to run BP, which uses the relationship between path-length and convergence as an optimality criteria. They contrast this choice of approach with standard information-theoretic approaches using a Chow-Liu tree to create a minimum spanning tree using a mutual information metric [31] by noting the computational complexity of computing a true information metric.

Chapter 3

DDF-SAM: DISTRIBUTED DATA FUSION THROUGH SHARING SUMMARIZED MAP INFORMATION

In this section, I introduce the DDF-SAM paradigm, as proposed in [34, 35, 36, 32, 33], for solving the Decentralized Data Fusion (DDF) problem using the Smoothing and Mapping (SAM) techniques developed for single-robot applications that is online, scalable, consistent and generalizable to a variety of sensing modalities. The primary structural assumption enabling distributed inference in DDF-SAM is that individual robots will only need to share a subset of the locally observed variables with neighboring robots, which leads to an architecture in which individual robots perform local SLAM, *summarize* their locally-optimized map information into a map with information only on these shared variables, and then transmit this summarized map to neighboring robots for fusion.

This chapter introduces the core contribution presented in this dissertation: the DDF-SAM paradigm for decentralized multi-robot SLAM. The claims for the approach are as follows:

1. DDF-SAM enables consistent, decentralized inference under the network constraints imposed by the DDF requirements through a process of sharing and caching summarized information between neighboring robots.

2. Choosing a subset of variables to share with the robot network for summarization exposes a design parameter enabling flexibility for different application domains.

3. By restricting the size of fusion neighborhoods, DDF-SAM manages scalability

Algorithm 1 General DDF-SAM.

initialize: for each robot r

$\Theta^r \leftarrow \{x^r_{init}\}$ ▷ initialize robot r at local origin
$\Phi^r \leftarrow \emptyset$ ▷ local factor graph
$\mathcal{M}^r \leftarrow \emptyset$ ▷ local summarized map cache

for each robot r at each discrete timestep k

$(\Phi^r, \Theta^r) \leftarrow processLocalMeasurements\,(\Phi^r, \Theta^r, k)$
$\Theta^r_s \leftarrow chooseSharedVariables\,(\Theta^r)$
if k is a summarization interval and $|\Theta^r_s| > d_{min}$
$\boldsymbol{m}^r_k \leftarrow summarize\,(\Phi^r, \Theta^r, \Theta^r_s, k)$
add $\boldsymbol{m}^r_k$ to cache $\mathcal{M}^r$ and any replace older $\boldsymbol{m}^r_{k-1}$
$(\mathcal{M}^r, \Phi^r, \Theta^r) \leftarrow communicateMaps\,(\mathcal{M}^r, \Phi^r, \Theta^r)$

with increasing numbers of robots by bounding the optimization problem and message size.

The formulation extends a local SLAM approach with the addition of neighborhood information such that I extend the local sensor horizon of each robot, while targeting the requirements for the system of being online, scalable and consistent. Algorithm 1 details a general formulation for the approach, which further sections will elaborate upon.

This chapter introduces the core motivations for the approach as well as the communication model, with specific implementations DDF-SAM 1.0, DDF-SAM 2.0 and Nonlinear DDF-SAM 2.0 in Chapters 4, 5 and 6, respectively, followed by a more general discussion of summarization techniques in Chapter 7.

3.1 Processing Local Measurements

At the base of the multi-robot inference system in DDF-SAM is a single-robot SAM process that enables each robot to build a local SLAM solution using only its locally available measurements. This section presents a reference formulation for single robot SLAM using batch smoothing designed to highlight the requirements of DDF-SAM

Algorithm 2 Generic local measurement processing stage for DDF-SAM

```
function processLocalMeasurements(Φ^r, Θ^r, k) :
    collect measurements Z_k^r from local sensors
    create local factors Φ_k^r from Z_k^r
    for each new factor φ_j^r ∈ Φ_k^r
        for each variable θ_i^r ∈ variables(φ_j^r)
            if θ_i^r ∉ Θ^r
                z_j^r ← measurement(θ_i^r)
                θ_i^r ← initializeVariable(Θ^r, z_j^r)
                add initialization θ_i^r to Θ^r
        add factor φ_j^r to Φ^r
    Θ^r ← optimize(Φ^r, Θ^r)
    return (Φ^r, Θ^r)
```

on a robot's local mapping capabilities. Each robot $r \in \mathcal{R}$ should be able to perform SLAM using sensor measurements Z^r from on-board sensors. While DDF-SAM does not depend strictly on the sensing modalities used, is necessary that there are variables relevant to neighboring robots, whether they be landmarks in the environment or scans associated with a given robot pose.

Algorithm 2 describes a generic formulation for single-robot SAM under the assumption that data associations are known and all variables can be initialized when first observed. In this representation, I maintain the density $p(\Theta^r|Z^r)$ as a combination of a factor graph Φ^r and a current estimate Θ^r on a given robot, and I perform an by adding a set of newly observed factors Φ_k^r to the graph and performing nonlinear optimization to yield a new estimate. In this formulation, I generalize initialization of each new variable $\theta_i^r \leftarrow initializeVariable\left(\Theta^r, z_j^r\right)$ to be possible using the current estimate Θ^r and the measurement z_j^r from a single factor, which for a full implementation is dependent on the measurement modality.

As a future notation for accessing portions of complex entities, such as a factor $\phi_j(\Theta_j; z_j)$, which contains both a list of affected variables Θ_j and a measurement z_j, I

will denote accessing specific portions with functions, such that $\Theta_j \leftarrow \mathit{variables}(\phi_j)$ and $z_j \leftarrow \mathit{measurement}(\phi_j)$. Similar notation will be used for different entities throughout this document. As a further simplification of notation, I will use Θ to denote both a listing of variable identifiers and the set of variables themselves, as they are typically identical. Cases where the distinction is significant will be noted explicitly.

The easiest choice of local SLAM modalities for use with DDF-SAM uses landmarks in the environment as a representation of the environment itself. I typically choose this approach for two reasons: 1) landmarks can be readily re-observed by neighboring robots, thus making fusion across platforms easier, and 2) landmark measurements from different robots can be considered statistically independent, which simplifies consistency constraints by only needing to track one source robot for a given measurement.

Because observations that can fix a robot in a global reference frame, such through GPS measurements or localization against a known map, are frequently unavailable due to environmental conditions, each robot performs local SLAM in its own local reference frame. While the origin with a local reference frame is arbitrary, I choose the first pose of every robot as the origin of its reference frame for convenience. The relative reference frame between robots, denoted by T_A^B as the transform from the frame of robot A to robot B, are not considered know to local robots and will need to be estimated online.

While this document primarily details a landmark-SLAM approach for decentralizing SAM, it is possible to use PoseSLAM variations employing scan-matching between robots. The complication that occurs in PoseSLAM measurements, however, is that a scan-match between two robots is associated with two separate sensor systems, and ensuring statistical independence of measurements is more difficult.

3.2 Summarizing Local Map Information

The core intuition behind the DDF-SAM approach is that while robots need to share information with neighbors as a part of collaborative SLAM, it is not necessary that all locally observed variables Θ^r from robot r will necessarily be useful to neighboring robots. Rather than sending its entire local state $p(\Theta^r)$ to its neighbors, I instead create a *summarization* of the locally observed map information which is over a smaller set of *shared variables* Θ^r_s (with corresponding Θ^r_u denoting unshared variables) chosen to be both useful to neighboring robots and to grow slowly as a set. In the overview of the approach in Algorithm 1, I update the set of shared variables anytime new map information is received from the local sensors with $\Theta^r_s \leftarrow chooseSharedVariables(\Theta^r)$. As a simplifying assumption, any variable $\theta^r_i \in \Theta^r$ chosen to be shared will always be shared in future timesteps, with the implication that I will never decide to stop sharing a particular variable.

While a simple approach to sharing map information between robots might simply send the values of the variables Θ^r themselves to neighboring robots, this approach would not capture the uncertainty still existing in a map being built online. By sharing a joint distribution $p(\Theta^r_s) \triangleq \left(\tilde{\Phi}^r_s, \Theta^r_s\right)$, it is possible for neighboring robots to fuse measurements in a consistent manner accounting for belief uncertainty.

Let the full summarized map structure $\boldsymbol{m}^r_k$, constructed from a local system in Algorithm 3, for the robot r at timestep k be the tuple $\left(\tilde{\Phi}^r, \Theta^r_s\right)$. Let $\tilde{\Phi}^r_s$ be a linearized graph that has been summarized to operate only on the set of saved variables Θ^r_s while only dependent on the locally observed measurements $Z^r_{1:k}$ as of time k, and Θ^r_s is the set of shared variables as well as the linearization point for $\tilde{\Phi}^r_s$. A significant consideration in the approach used via Algorithm 3 is that before elimination, it is necessary to linearize the full graph Φ^r around its current estimate Θ^r in order to perform elimination, because elimination is not possible on arbitrary nonlinear factors.

Algorithm 3 Summarization via partial elimination from

`function` $summarize\left(\Phi^r, \Theta^r, \Theta_s^r, k\right)$:

$\tilde{\Phi}^r \leftarrow linearize\left(\Phi^r, \Theta^r\right)$ ▷ $p\left(\Theta^r; Z_{1:k}^r\right)$

$\tilde{\Phi}_{summarized}^r \leftarrow summarizeLinear\left(\tilde{\Phi}^r, \Theta_s^r\right)$ ▷ $p\left(\Theta_s^r\right)$

$\boldsymbol{m}_k^r \leftarrow createSummarizedMap\left(\tilde{\Phi}_{summarized}^r, \Theta_s^r, k\right)$

`return` $\boldsymbol{m}_k^r$

I claim that sending summarized densities to neighboring robots will produce a decentralized mapping system that is more computationally scalable. If robots were to share all of their raw measurements with neighboring robots, any given platform would need to perform a larger, nonlinear data fusion problem on all variables (both poses and landmarks). This system will continue to grow over time due to each robot adding more poses. Sending summarized map information, however, corresponds to distributing the computational workload between each robot, such that each platform performs the nonlinear optimization necessary to make a consistent local map, yielding a better initialization to neighboring robots. Furthermore, summarization reduces the size of the data fusion problem on each platform to only grow at the rate at which robots observe new landmarks, thus keeping the data fusion problem small.

The key requirement for consistency, insofar as the avoidance of double-counted information when fusing summarized maps on neighboring robots(see Section 2.6.2), is that the summarized map $\boldsymbol{m}_k^r$ *is dependent on exactly its locally available measurements* Z^r, such that there is no contribution of neighboring robots. By satisfying this requirement, I can consider summarized maps received from different robots to include statistically independent observations and fuse these measurements without becoming over-confident due to double-counted information.

Algorithm 4 Exact summarization of a linear system $\tilde{\Phi}$ around a given set of variables Θ_s using partial elimination to perform marginalization.

`function` $summarizeLinear\left(\tilde{\Phi}^r, \Theta_s^r\right)$:

`compute variable ordering` $\mathcal{P}$ `of` $\tilde{\Phi}^r$ `such that` Θ_s^r `are last`
$\tilde{\Phi}^r \leftarrow reorderVariables\left(\tilde{\Phi}^r, \mathcal{P}\right)$
$\left(\mathcal{B}, \tilde{\Phi}_{summarized}\right) \leftarrow multifrontalEliminatePartial\left(\tilde{\Phi}^r, \Theta_u^r\right)$
`return` $\tilde{\Phi}^r_{summarized}$ $\triangleright\ p\left(\Theta_s^r\right)$

3.2.1 Summarization via Partial Elimination

The summarization process for computing a summarized map for a robot r, starting from a local SAM solution expressing $p\left(\Theta^r\right)$ and yielding a summarized density $p\left(\Theta_s^r\right)$, is a marginalization procedure that is a natural extension of the elimination algorithm (see Section 2.5.2). Given a $p\left(\Theta^r\right)$ expressed as a factor graph $\Phi^r\left(\Theta^r\right)$, I eliminate each variable $\theta^r \in \Theta_u^r$, which factorizes the graph into a density $p\left(\Theta_u^r|\Theta_s^r\right)p\left(\Theta_s^r\right)$. This partial elimination procedure produces an exact marginal over the shared variable set, based on all locally available measurements Z^r.

The standard approach for summarization of a graphical system appears in Algorithm 4, which uses multifrontal partial elimination to factorize the system into $p\left(\Theta_u^r|\Theta_s^r\right)p\left(\Theta_s^r\right)$, expressed as the tuple $\left(\mathcal{B}, \tilde{\Phi}_{summarized}\right)$, where the Bayes tree $\mathcal{B}$ encode $p\left(\Theta_u^r|\Theta_s^r\right)$ and the remaining factors $\tilde{\Phi}_{summarized}$ encode $p\left(\Theta_s^r\right)$. Computationally, this factorization problem is typically quite dense because the variable reordering enforced by $\mathcal{P}$ yields a large degree of fill-in on the shared variables, frequently resulting in $\tilde{\Phi}_{summarized}$ being a single dense linear factor.

3.3 Communicating Summarized Maps

The communication phase of DDF-SAM combines distributing cached summarized maps to each neighboring robot with receiving of summarized maps through pairwise

Algorithm 5 Generic communication phase for DDF-SAM for a robot r. A message from robot a to b is a set of summarized maps $\mathcal{M}^{a \to b}$. This version uses a representation of the estimate to update with summarized information as a factor graph Φ and a current estimate Θ, though other representations of state can be used as well, such as in the case of incremental formulations.

```
function communicateMaps(M^r, Φ, Θ):
    determine communication neighborhood N_r
    for each robot a ∈ N_r
        M^{a→r} ← twoPhaseSharing(a, M^r)   ▷ Maps from a
        for each summarized map m_j^b ∈ M^{a→r}
            (Φ, Θ, M^r) ←
            addSummarizedMap(m_j^b, Φ, Θ, M^r)
    return (M^r, Φ, Θ)
```

communication. In this process, formulated in Algorithm 5, each robot will share information with the other robots in its communication neighborhood N_r, resulting a robot r receiving a set of summarized maps as a message $\mathcal{M}^{a \to r}$ from a robot a.

I make two claims about the communication approach used DDF-SAM:

- *Indirect Information Propagation*: Because robots share sets of maps, rather than only the latest summarized map $\boldsymbol{m}_k^a$ from robot a, information will propagate throughout the network *indirectly*. Furthermore, if a robot becomes disabled, its most recent summarized map can still exist in the caches of neighboring robots. This allows DDF-SAM to be robust to communication and node failures when scaling to larger teams.

- *Fusion Neighborhood Bounding*: By bounding the number of robots any given robot r will fuse information from to a constant K, I can bound the computational and communication cost as the number of robots increases, enabling online operation. By limiting the size of the multi-robot fusion problem on each platform, I avoid each robot acting forming a full global map.

The remainder of this section introduced the communication model for DDF-SAM,

with the message passing approach in Section 3.3.1 and the fusing of a single summarized map in Section 3.3.2. Sections 3.3.3 and 3.3.4 detail the basis for the communication claims of indirect information propagation and fusion neighborhood bounding, respectively.

3.3.1 Two-Phase Synchronous Communication

The communication model for DDF-SAM is pairwise, symmetric, and opportunistic - a model I designed to minimize message-passing constraints placed on individual agents and be easily generalizable. As in the standard DDF networking model, all communication is *pairwise* in which a robot r may communicate directly with robots in its neighborhood N_r, and any indirect communication occurs through pairwise messages. I require that communications are *symmetric* in nature in order to avoid implicit hierarchies or bottlenecks on the network, and to allow additional robots to be added or removed from the system. To further ensure that this is a decentralized network, this approach uses *opportunistic* communication, in which robots will attempt communication with any neighbors within range, without any a priori decisions on network structure.

The simplest communication model for each robot r sharing their entire caches $\mathcal{M}^r$ with each neighbor, but this will result in sending redundant data. The most obvious case of redundant data transmission is when a robot a sends a summarized map $\boldsymbol{m}_j^r$ to another robot b has cached $\boldsymbol{m}_k^r$, where k is a later timestep than j. In this instance, the message $\mathcal{M}^{a\to b}$ contains a summarized map that robot b will ignore completely.

The two-phase cache sharing model, detailed in Algorithm 6, addresses the problem of updating caches by splitting communication into sharing cache listings, and then sharing only relevant summarized maps. The listing messages $\bar{\mathcal{M}}^{r\to a}$ contain only the robot identifier and timestamp for each summarized map in $\mathcal{M}^r$, yielding

Algorithm 6 Two-phase symmetric communication between the local robot r and a neighboring robot a, where the first phase shares map listing messages $\bar{\mathcal{M}}^{a \to r}$ and the second phase shared summarized maps $\mathcal{M}^{a \to r}$.

```
function twoPhaseSharing(a, M^r) :
    M̄^{r→a} ← ∅;                                  ▷ Cache listing message
    for each m_k^b ∈ M^r where b ≠ a
        add cache state listing (b, k) to M̄^{r→a}
    transmit M̄^{r→a} to robot a
    receive message M̄^{a→r} from robot a
    M^{r→a} ← ∅;                                  ▷ Summarized map message
    for each (b, , j) ∈ M̄^{a→r}
        if there exists a m_k^b ∈ M^r and k > j
            add m_k^b to the message M^{r→a}
    transmit message M^{r→a} to robot a
    receive message M^{a→r} from robot a
    return M^{a→r}
```

a substantially smaller message than simply sending all of $\mathcal{M}^r$ initially. On receipt of $\bar{\mathcal{M}}^{a \to r}$, robot r can then assemble a message $\mathcal{M}^{r \to a}$ containing only cached maps that will be newer than those already in $\mathcal{M}^a$. At a minimum, $\mathcal{M}^{r \to a}$ will contain the latest $\boldsymbol{m}_k^r$, for which robot r will always have the latest version, up to a maximum size where $\mathcal{M}^{r \to a} = \mathcal{M}^r$.

3.3.2 Fusing a Summarized Map

The core step in the process of receiving summarized maps fusing a given summarized map $\boldsymbol{m}_j^b$ with a current system represented with a factor graph and estimate(Φ, Θ), which is outlined for a generic case in Algorithm 7. This step also manages updates to the cache $\mathcal{M}^r$ under two key constraints: (1) replace older summarized maps with newer ones, such that the cache always has the most recent version, and (2) limit the size of the cache to a fixed bound K, such that I will not attempt to add maps from additional neighboring robots once $|\mathcal{M}^r| = K$. Keeping only the most recent summarized maps avoids storing unnecessary information, since each summarized

Algorithm 7 Generic adding an incoming summarized map $\boldsymbol{m}_j^b$ on a robot r summarized at time j, with the same generic representation of the estimate state as in Algorithm 5.

```
function addSummarizedMap(m_j^b, Φ, Θ, M^r)
    find any previously observed m_k^b ∈ M^r
    if timestamp j > k or |M^r| < K

        fuse summarized map m_j^b with current estimate (Φ, Θ)
        if map fusion successful

            replace any previous m_k^b in M^r with m_j^b
    return (Φ, Θ, M^r)
```

map has strictly more information than its predecessors.

The actual sensor fusion step of the process, which updates a current system (Φ, Θ) with the map information $\left(\tilde{\Phi}^b, \Theta_s^b\right)$ from the summarized map $\boldsymbol{m}_j^b$, varies significantly between DDF-SAM approaches due to (a) the need to avoid information double-counting due to mixing local and neighborhood information and (b) the impact of linearizing each summarized map in a local reference frame. If I were to naively add a summarized map $\boldsymbol{m}_j^b$ to the local system for a given robot (Φ^r, Θ^r) and then perform summarization directly, the resulting summarized map $\boldsymbol{m}_k^r$ would contain information contributed by neighboring robots, resulting in double-counted measurements were to robot b to attempt to fuse its map. Furthermore, there is no straightforward way of performing standard optimization over a system combining both locally observed factors with full nonlinear measurement models and linearized factors that were originally linearized in a separate and unknown reference frame.

3.3.3 Information Propagation

In the unbounded neighborhood size case, I allow each robot to fuse maps received from all other robots over time, which will converge to a global map given sufficient

time so long as the union over the network graph over time is connected. As a point of clarification for the analysis, the communication neighborhood N_r of a robot r refers to exactly those robots in direct communication at a given time, and I will consider the set of summarized maps on each robot $\mathcal{M}^r$ as a set of "fusion neighbors" that accumulate over time. I derive this property as an extension of the convergence proofs in consensus systems (See Section 2.6.3.1), in which a communication network (even with random switching) will converge using only pairwise communication. In the case transmitting cached data indirectly between robots, I can effectively introduce more connections in the network union graph over time, thus improving the convergence rate. If I consider a graph of fusion neighbors $\mathcal{G}_f \triangleq (\mathcal{V}, \mathcal{E})$, where each robot is a vertex $r \in \mathcal{V}$ and an edge $e_{ab} \in \mathcal{E}$ exists between robots a and b if summarized maps exist in each robot's cache such that $\boldsymbol{m}^a \in \mathcal{M}^b$ and $\boldsymbol{m}^b \in \mathcal{M}^a$, global convergence occurs when the graph $\mathcal{G}_f$ is complete. This cached propagation of information through the network affords several advantages in resiliency to network topology changes: (a) in the event of node failure, the latest shared summarized map will still be present in caches in the network, and (b) robots can update their neighborhood maps at any point in the process using currently cached information and will not have to wait for synchronized messages from multiple robots.

3.3.4 Fusion Neighborhood Bounding

While this convergence result is significant in maintaining a robust decentralized inference solution, converging to a global map on each platform will become intractable as the number of robots in the system increases, motivating a the use of a bound K on the number of fusion neighbors $\mathcal{M}^r$ for a given platform. In terms of the fusion graph $\mathcal{G}_f$, K is a bound on the degree of a node. More significantly, it acts a bound on the size of the local fusion problem for any given platform, which ensures that once a robot has reached the bound$|\mathcal{M}^r| = K$, the computational requirements for updating

the map solution only grow as the size of individual summarized maps $\boldsymbol{m}^a \in \mathcal{M}^r$ grow.

This bound yields a significant capability of the overall distributed inference system: when robots are not observing new landmarks, the size of neighborhood graphs remains bounded by a factor of K. Adjusting K corresponds to increasing the effective sensor horizon of a given robot at the expense of computation and communication. With a good choice of shared variables, as described in Section 3.2, such as fixed environmental landmarks, it is possible that as robots transition from primarily exploring (encountering new landmarks) to patrolling (re-observing old landmarks) that the size of individual summarized maps will become constant, yielding constant time map updates.

3.4 Evaluating DDF-SAM

I validate DDF-SAM experimentally through performance and correctness metrics which measure the ability of the system to operate as an online SLAM system that is scalable to large scale teams while ensuring a consistent solution on each platform. The primary performance metrics measure the computation time for each phase in DDF-SAM, and communication bandwidth between the robots, measured in size of messages. I use estimation error as a metric for correctness of the solutions maintained by the system. To evaluate the effect of bounding the size of each robot's neighborhood estimation problem to a constant K, I run experiments for multiple values of K. The remainder of this section details the evaluation approach and the experimental setup.

3.4.1 Comparison Implementations

In addition to the different configurations of DDF-SAM presented in following chapters, I compare to single-robot SLAM and centralized multi-robot SLAM, which can be considered low and high water marks, respectively, for a multi-robot system. For

each of these cases, I use both a state-of-the-art incremental SAM solver (iSAM 2.0, as described in [85]) and a nonlinear batch SAM solver using Levenberg-Marquardt optimization (as described in [42]). In the case of the batch solver in an incremental setting (where new measurements come in every timestep k), I re-run optimization over the full problem at each timestep k, using the solution from the previous timestep $k-1$ as an initial estimate for the new problem.

As the baseline expectation for any multi-robot SLAM system, I run a standard SAM solver on the locally available measurements, with no observations or messages received from neighboring robots. This approach uses strictly less measurement information than a technique incorporating information from other robots, which should result in greater estimation uncertainty in the solution.

As an upper bound on estimation quality, I compare the DDF-SAM approaches to a straightforward centralized multi-robot SLAM approach that works under a perfect communication assumption. This solver receives raw measurements from each robot and combines them into a single large factor graph over all variables, including the complete trajectories of each robot and the entire map. This technique acts as an upper bound over techniques such as DDF-SAM, as it can optimize the complete nonlinear optimization problem without any approximations induced through linearizations or incomplete maps.

3.4.2 Experimental Setup

I performed experimental validation of the primary DDF-SAM claims of online, scalable and consistent operation through an offline test suite comparing DDF-SAM variations and control implementations. Because these experiments evaluate a variety of approaches, I use an offline benchmark that processes a stored dataset and updates each solver configuration under test, while also collecting statistics for evaluation metrics. While the experimental setup is designed to be representative of multi-robot

SLAM systems in the field, I make some systematic simplifications to make differences in inference approaches more clear, as well as ensuring that aggregate statistics collected across a large number of robots are meaningful.

The largest simplification is the use of known, globally consistent data associations for each landmark observed, which decouples the SLAM front-end challenges present in real-world systems from the inference techniques covered in this work.

The other key simplification over a real-world system is in synchronization of timing of operations, particularly updates and communication, which is necessary for computing aggregate statistics across a large number of robots with different configurations. For local updates, this is representative of having fixed-rate sensor systems. However, for message passing, the communication phase occurs at a fixed interval on each robot, such that all robots distribute and fuse maps simultaneously. To simulate the effect of messages coming at different times, I randomize the order in which robots share information across a network graph.

As a special case of synchronized communication for the centralized multi-robot SLAM implementation, I maintain the same update interval for sharing of information where each robot sends all measurements since the previous transmission back to the local solver.

3.4.3 Computational Performance Metrics

In order to validate claims of both online operation and scalability to large teams of robots, I measure computational performance for each of the key steps in the DDF-SAM process: local updates, map summarization and neighborhood updates. Local update timing is the time to incorporate newly observed local sensor data, exactly as in a single-robot SLAM context, with an update at every timestep. Summarization timings are computed any time a summarization event occurs, which under typical test conditions used for evaluation, are at a fixed interval over a number of timesteps.

Measuring the time for fusing neighborhood maps is split into a per-message update time and a total neighborhood update time, which includes a group of summarized maps. For a typical case in which a robot has several incoming summarized maps for neighboring robots, resulting in several separate neighborhood updates. Because the number of incoming summarized maps will depend on the size of the neighborhood, I measure both single-map and full neighborhood updates to expose the effect of network graph connectivity.

3.4.4 Communication Bandwidth

To validate the ability of the system to scale to large teams of robots, particularly under limited communication, I measure the communication bandwidth use, through a proxy metric of the size of summarized map messages $\mathcal{M}^{a \to b}$ between robots in pairwise communication. To avoid dependence in evaluation on implementation-specific size metrics, such as through various data structure serialization techniques which can introduce unnecessary overhead costs, I measure the size of messages as a lower-bound on how many numbers are necessary for lossless encoding.

3.4.4.1 Summarized Map Transmission Size

For a summarized map $\boldsymbol{m}_k^r$ encoded by a factor graph $\tilde{\Phi}_k^r$ of size N, a set of variables Θ_s^r of size M, as well as a timestamp k and an robot identifier r, the transmission size $s\left(\boldsymbol{m}_k^r\right)$ is given by

$$s\left(\boldsymbol{m}_k^r\right) \triangleq 2 + s\left(\Theta_s^r\right) + \sum_{j=1}^{M} s\left(\tilde{\phi}_j\right), \tag{7}$$

$$s\left(\Theta_s^r\right) \triangleq N + \sum_{i=1}^{N} \dim\left(\theta_i\right), \tag{8}$$

where the size $s\left(\tilde{\phi}_j\right)$ of a linear factor depends on its structure. In these formulations, timestamp k, the robot identifier r and each of the variable identifiers in Θ_s^r are all alloted a single number counting towards the transmission size. The dimension of

a variable $\dim(\theta_i)$ is the size of the tangent space, as used for linearization, which for typical variables ranges from 2 for a planar landmark $l_i \in \mathbb{R}^2$ to 6 for a full pose $x_i \in \mathcal{SE}(3)$.

For a linear factor in Jacobian form $\tilde{\phi}_j = \|A\delta - b\|^2$ (which corresponds to square-root information form), where the involved variables are given by Θ_j, with N_j variables and $b \in \mathbb{R}^m$, the transmission size $s\left(\tilde{\phi}_j\right)$ is

$$s\left(\tilde{\phi}_j\right) \triangleq N_j + m\left(\sum_{i=0}^{N_j} \dim(\theta_i) + 1\right), \tag{9}$$

which counts all entries of A. For factorized forms, such that there are either structural zeros (as in the case of a factor based on a conditional in the form $\|R\delta - d\|^2$, where R is upper triangular, I count only the nonzero entries in the matrix. For a linear factor specified in Hessian form $\tilde{\phi}_j = \frac{1}{2}\delta^T G\delta - \delta^T g + \frac{1}{2}f$, where $G \in \mathbb{R}^{n\times n}$ is symmetric, $g \in \mathbb{R}^n$, $f \in \mathbb{R}$ and $n \triangleq \dim\Theta_j$, the transmission size is

$$s\left(\tilde{\phi}_j\right) \triangleq N_j + \frac{1}{2}(n+2)(n+1), \tag{10}$$

which accounts for only the upper triangular portion of the symmetric matrix G.

3.4.4.2 Network Traffic

Given a transmission size measure for individual summarized maps $\boldsymbol{m}_k^r$, it is possible to measure the network cost of a single pairwise sharing operation between robots a and b as

$$s(a \leftrightarrow b) \triangleq s\left(\mathcal{M}^{a\to b}\right) + s\left(\mathcal{M}^{b\to a}\right). \tag{11}$$

To measure communication costs at a given node r during a sharing operation with its neighborhood N_r, I can measure the total traffic at the node

$$s(r \leftrightarrow N_r) \triangleq \sum^{N_r} s(r \leftrightarrow a). \tag{12}$$

These metrics provide a means to evaluate the communication costs associated with the approach, as well as isolating bottlenecks in the approach.

3.4.4.3 Ideal Centralized Communication

For the centralized scenario, I model communication as all robots sending all raw measurements $Z^r_{j:k}$ accumulated over a time interval from j to k to the fusion center using direction network connections. I use measurements accumulated over intervals, rather than a separate transmission at each timestep to evaluate the effect of less frequent communication at the same intervals as the robots transfer data.

3.4.5 Estimation Error

I evaluate consistency claims in all DDF-SAM cases by measuring the error in the map solutions computed, with metrics for both the error in the trajectory of the robot and the state of the map when compared to ground truth. The separate reference frames of each robot complicates measurement of exact error against a ground truth frame, in which a given solution computed on a robot may contain variables in different frames of reference.

3.4.5.1 Trajectory Error

In the case of trajectory error, I use an incremental error parametrization, in which the total error for a given trajectory X^A length N compared to a reference trajectory X^r

$$\sum_{i=0}^{N-1} \left\| x_i^A \ominus x_{i+1}^A - x_i^r \ominus x_{i+1}^r \right\|$$

where $\ominus$ is a manifold between operation (see Section 2.1.1 for definition) , which is invariant to absolute reference frame of the trajectory. This incremental error metric has also been used by others [92] in SLAM systems, not only to account for different reference frames, but also to reduce the error induced by lever-arm effects by a small orientation error early in a trajectory.

As a detail for evaluating error in terms of poses, I split the error into a rotation and translation component and present these errors separately. This separation is

necessary because the possible rotation error is limited to a $(-\pi, \pi)$ range and a combined error term can become dominated by error in translation.

3.4.5.2 Landmark Error

I specify landmark error in terms of an error in a constellation of landmarks L^r from robot r, such that if there is a ground truth landmark map $\hat{L}$ in a reference frame g, I compute total error as

$$\left\| \hat{L} - T_r^g \circ L^r \right\|^2, \tag{13}$$

where T_r^g is a transform from the reference frame of r back to the ground truth frame g. One challenge in using this approach for measuring error is determining the value of the transform T_r^g to perform this estimate. The simplest approach is to use the ground truth trajectories of the robots, where the first pose x_0^r defines the local origin in the reference frame r, so the ground truth first pose $\hat{x}_0^r$ is T_r^g. To account for the possibility that there is estimation error affecting the first poses of the robot, as would happen if near initialization the robot did not observe any landmarks, I also compute a T_r^g by computing the best-case transform through an alignment via SVD of $\hat{L}$ and L^r. With this reference frame transform, the error in (13) is a measure of constellation alignment. To allow for comparisons in estimation error between maps with differing numbers of landmarks, I typically normalize error by the number of landmarks present in a given map, yielding a measurement of the mean error for a single landmark in a given map estimate.

3.4.6 Datasets

By evaluating DDF-SAM through a series of simulated and real-world datasets, I validate claims in this document, specifically focusing on ensuring the system is online, scalable and consistent. In order to evaluate over large scale systems, my primary datasets are simulated environments populated with landmarks and robot trajectories. Because the simulation has exact ground truth for both the solution values and

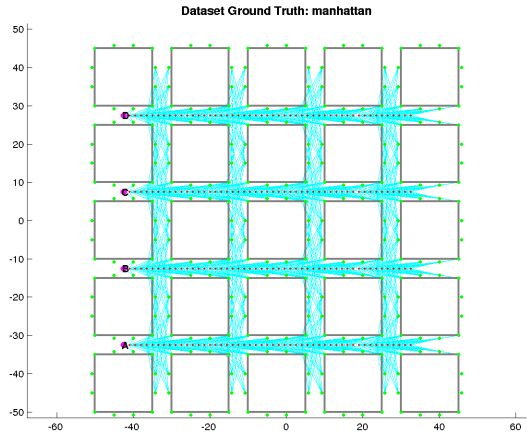


Figure 8: Manhattan-world simulated dataset with 4 robots, showing robot trajectories flying left to right and observations of landmarks along the sides and corners of buildings. In this case, robots have a 180 degree field of view and are able to communicate with robots on their neighboring street.

data associations for all measurements, it provides a good basis for changing experimental conditions. I also use real-world datasets collected using small teams of robots to demonstrate that the system is robust to realistic conditions.

3.4.6.1 Manhattan-world Simulation

As a small example for illustrative purposes, I use a simple planar Manhattan-world example, with ground truth illustrated in Figure 8, in which a small team of robots flies down streets between a series of line-of-sight blocking buildings. In this scenario, the goal is to extend the sensor horizon of individual robot to enable both further navigation, as well as improving certainty on their maps by combining both local and neighborhood measurements. Because individual robots never close any loops during the course of linear trajectories, the uncertainty of the estimate for poses and landmarks will increase without bound, but through the addition of measurement information from neighboring robots, I can reduce this uncertainty.

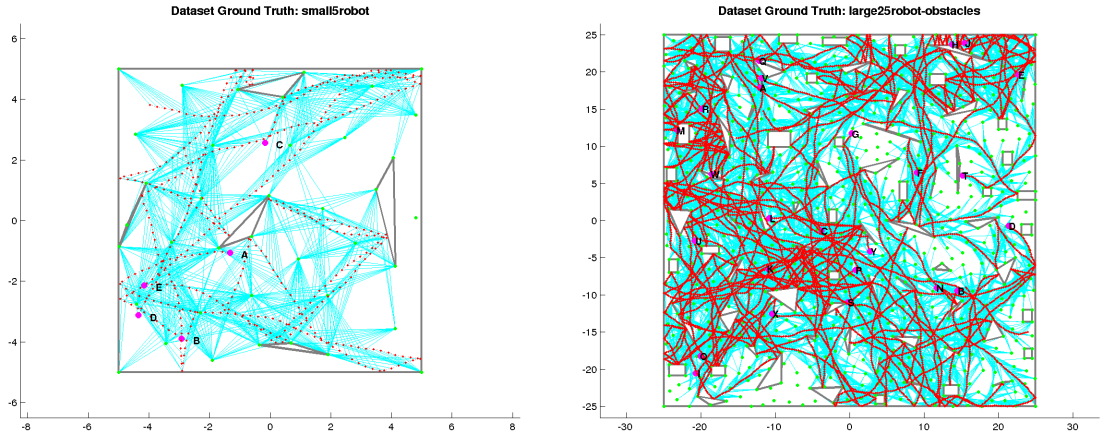


Figure 9: Example environments produced by the large scale planar simulation environment, showing both a small example for testing (left) and a larger example for evaluation. In both cases, walls are denoted with blue lines, landmarks are green dots, and robots have red trajectories. The trajectories in each case are random walks at constant speed and deflecting off of obstacles.

3.4.6.2 Large Scale Simulation

I developed a simulator able to handle SLAM with a large number of robots with randomly generated environments incorporating line-of-sight blocking obstacles, fixed landmarks, and random walk behavior for the robots. By using a randomly generated scenario, it is possible to adjust the scale of the environment to evaluate different conditions. This simulation environment is based on work completed in [35, 36], with additional statistics reporting. Figure 9 shows two examples of scenarios created in this simulation.

3.4.6.3 Freiburg Dataset

Collaborators at the University of Freiburg performed experiments using a heterogeneous team of three planar robots (see Figure 10), conducted in the parking lot of the computer science campus in Freiburg [35]. Since the environment does not contain a sufficient amount of detectable features, poles were placed through the area (see Figure 10 for an example pole in comparison to the robots, and Figure 11 for

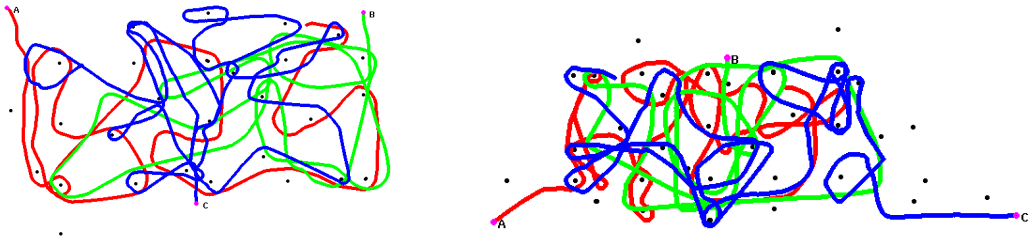
Figure 10: The robots used for the Freiburg dataset, consisting of an ActiveMedia Pioneer2, Pioneer2 AT and a PowerBot, each equipped with a SICK LMS 291 laser range finder used primarily to detect pole features (shown at right).

Figure 11: The parking lot used for experiments, shown with poles placed (left), and aerial view of the area (center). For an approximation of ground truth, I use a batch centralized solution for run 1, showing the output (right) of GMapping [61, 62] applied to the full laser scans from the robots, with trajectories of the three robots overlaid and color coded.

the full array of landmarks). Bearing-Range feature measurements came from a pole detector and were filtered in post-processing to remove noisy features, such as the legs of the operators.

There were two runs conducted where the robots were moved in the environment for about 20 min, with trajectory lengths ranging from 9,000 to 10,000 poses, covering an area around 100 meters long. The trajectories of the robots during the first run can be seen in Figure 11. While there is no true ground truth for this experiment, it is possible to reconstruct the environment using the full laser scans, rather than just the detected features, and overlay robot trajectories, as shown in Figure 11.

For the purposes of evaluation, I used a centralized version of the system as a ground truth, as shown in Figure 12, which combined both local and between-robot data associations in a single solution. In this case, I ran the data from each robot through a standard smoothing algorithm to get a single-robot map solution. The



(a) Run 1 (b) Run 2

Figure 12: Centralized refined solutions for the Freiburg multi-robot datasets, created as a result of global optimization and data associations constructed from post-processing. Trajectories correspond to robots A, B, C, with colors red, green and blue, respectively, and the black dots are landmarks.

local map data association uses a simple nearest neighbor approach with fixed gates to determine whether to associate a new measurement with a specific known landmark or create a new landmark. After solving for the single robot maps with additional batch refinement of the solution, I computed between-robot data associations using a simple application of RANSAC to find the rigid transforms and correspondences between the robots. Using these correspondences and initial transforms, I performed global batch optimization to yield the final ground truth solutions.

It should be noted that these data associations are designed to be *conservative* in that decisions to associate a measurement with an observed landmark are designed to avoid adding a connection where none should exist. This prevents the more significant failure mode of data association failure, in which an outlier measurement substantially distorts the solution. The result of this conservative approach is that there will occasionally be duplicate landmarks which should map to a single ground-truth landmark, but appear in the map solution as multiple landmarks.

For the Freiburg datasets, due to their long length, I used less frequent communication between with a longer delay before the robots were allowed to begin communication. In this case, the robots had a forced 1000 timestep delay before

communication could occur, and only shared data every 100 timesteps. I also assumed perfect communication, so each robot could always talk to its neighbors.

3.5 Discussion

In this chapter, I introduced the basic architecture of the DDF-SAM approach for decentralized inference through distribution of summarized map information over a chosen subset of shared variables, which is enables online, scalable and consistent decentralized SAM. The system architecture ensures that the approach is an online inference approach that shares summarized map information across the network, allowing individual robots to fuse these summarized maps as necessary. The communication model ensures that the system is scalable with an increasing number of robots because any individual robot is only dependent on its local neighbors for map information, and by bounding the neighborhood size, I can bound the size of the inference problem on each platform. The key requirement for consistent estimates in DDF-SAM is the summarized maps are dependent on exactly the locally available measurements, which makes the tracking of information sources during neighborhood updates feasible.

While this general architecture provides the basis for communication and sharing of information for the remainder of the the work, the key remaining operation to implement DDF-SAM is a neighborhood update operation which actually fuses summarized maps. The following chapters present two different approaches for fusing neighborhood map information, with Chapter 4 introducing a simpler two-map batch optimization technique and Chapters 5 and 6 extending the approach to use single-map incremental SAM techniques.

Chapter 4

DDF-SAM 1.0: BATCH DECENTRALIZED SAM

In this chapter, I present DDF-SAM 1.0, which contributes a fully realized DDF-SAM approach that is online, scalable, consistent and generalizable to a variety of sensing modalities, validated with both simulated and real-world experimental results. DDF-SAM 1.0, as originally proposed in [34, 35], with further experimental evaluation in [36, 72], is an implementation of the DDF-SAM architecture designed around batch nonlinear optimization techniques for fusion of cached summarized maps into a consistent neighborhood map. The key structural characteristic of the approach is the means by which it prevents information double-counting, in that each robot maintains a separate local and neighborhood map designed to prevent map information from neighboring robots from affecting the local map and summarizations.

In this approach, I avoid double-counting by maintaining two separate systems, a pure neighborhood system $\left(\Phi^{N'_r}, \Theta^{N'_r}\right)$ and a pure local system (Φ^r, Θ^r). In this notation for the neighborhood system, I use N'_r for convenience, where $N'_r \triangleq \{N_r, r\}$. The *only* connection between these systems is through adding the local summarized map $\boldsymbol{m}_k^r$ to the neighborhood system - thereby preventing information contributions from other robots affecting the locally generated summarized map $\boldsymbol{m}_k^r$ and causing overconfidence due to information double-counting.

4.1 Processing Local Measurements

Because DDF-SAM performs local SAM completely isolated from information received from other robots, the local measurement processing stage is equivalent to a single-robot system, as described in Algorithm 2. While there are a variety of approaches to solving this local system, I will present a straightforward Gauss-Newton

Algorithm 8 DDF-SAM 1.0 overview, which extends the generic DDF-SAM framework of Algorithm 1 through the use of a separate neighborhood estimate $\left(\Phi^{N'_r}, \Theta^{N'_r}\right)$ in addition to a pure local system (Φ^r, Θ^r) which is never updated with information from summarized maps.

`initialize: for each robot` r

$\Theta^r \leftarrow \{x^r_{init}\}$ ▷ `initialize robot` r `at local origin`
$\Phi^r \leftarrow \emptyset$ ▷ `local factor graph`
$\Theta^{N'_r} \leftarrow \emptyset$; $\Phi^{N'_r} \leftarrow \emptyset$; ▷ `Empty neighborhood system`
$\mathcal{M}^r \leftarrow \emptyset$ ▷ `local summarized map cache`

`for each robot` r `at each discrete timestep` k

$(\Phi^r, \Theta^r) \leftarrow processLocalMeasurements\,(\Phi^r, \Theta^r, k)$
$\Theta^r_s \leftarrow chooseSharedVariables\,(\Theta^r)$
`if` k `is a summarization interval and` $|\Theta^r_s| > d_{min}$
$\boldsymbol{m}^r_k \leftarrow summarize\,(\Phi^r, \Theta^r, \Theta^r_s, k)$
`add` $\boldsymbol{m}^r_k$ `to cache` $\mathcal{M}^r$ `and any replace older` $\boldsymbol{m}^r_{k-1}$
$\left(\mathcal{M}^r, \Phi^{N'_r}, \Theta^{N'_r}\right) \leftarrow$
$communicateMaps\left(\mathcal{M}^r, \Phi^{N'_r}, \Theta^{N'_r}\right)$

batch optimization approach as an illustrative example. In practice, this local system can be implemented using incremental solvers, such as iSAM [85], or more sophisticated nonlinear optimization techniques such as Levenberg-Marquardt with trust-region based regularization to improve convergence.

Algorithm 9 details batch nonlinear optimization for a single system modeled by a factor graph Φ and with an initial estimate of the solution Θ_{init}. Note that the formulation shown is a relatively simple approach, and more details are available in Section 2.5.1. Of note in this formulation is the function $multifrontalEliminate\left(\tilde{\Phi}\right)$, which yields a Bayes tree through either QR or Cholesky factorization of the underlying sparse linear problem. The underlying linear algebra in these steps starts with a sparse linear system $A\Delta - b$ represented by the linearized graph $\tilde{\Phi}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, where n is the tangent space dimension of Θ and m is the number of constraints. The reordering step $\tilde{\Phi} \leftarrow reorderVariables\left(\tilde{\Phi}, \mathcal{P}\right)$ chooses an elimination ordering designed to minimize the fill-in during elimination, which has significant

Algorithm 9 Generic iterated nonlinear optimization with a direct multifrontal linear solver, representative of Gauss-Newton approaches without regularization.

```
function optimize (Φ, Θ_init):
    Θ ← Θ_init;
    compute minimum fill column ordering 𝒫 from Φ
    while Θ not converged
        Φ̃ ← linearize (Φ, Θ) ;                ▷ See equations (2,3)
        Φ̃ ← reorderVariables (Φ̃, 𝒫)
        ℬ ← multifrontalEliminate (Φ̃)          ▷ p (Θ; R, d)
        Δ ← backsubstitute (ℬ) ;                ▷ Δ ≜ R\d
        Θ ← Θ ⊕ Δ;                              ▷ Manifold retract
    return Θ;
```

effects on system performance [42, 3].

4.2 Summarizing Local Map Information

DDF-SAM 1.0 performs map summarization using batch partial elimination approach described in Section 3.2, using the generic summarization approach detailed in Algorithms 3 and 4 to construct the set of summarized factors $\tilde{\Phi}^r_{summarized} \propto p(\Theta^r_s)$ around the shared variables for the robot r, starting from the pure local factor graph Φ^r. The summarized map structure $\boldsymbol{m}^r_k$ for robot r at a time k contains linearization point for the shared variables Θ^r_s as well as the the summarized linear factors $\tilde{\Phi}^r_{summarized}$. Figure 13 illustrates map summarization for a simple robot example, showing the full local factor graph Φ^r and the summarized map where the shared variables Θ^r_s is the set of landmarks.

By summarizing the from a purely local factor graph Φ^r, which does not include any neighborhood contributions, this summarization technique is guaranteed to avoid double-counting information, thus meeting the consistency requirement for summarized maps.

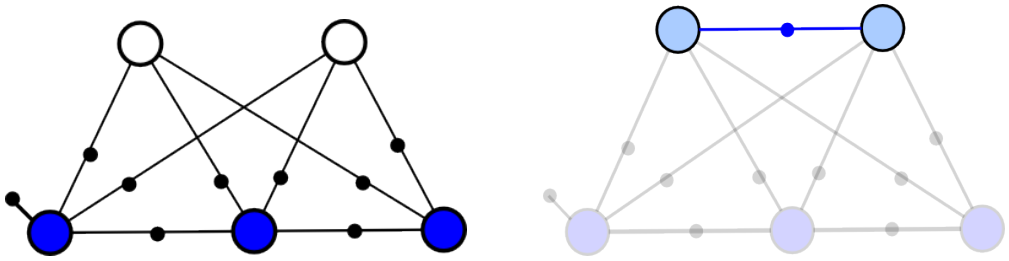


Figure 13: Summarization example for single robot case, showing a robot with three poses (blue circles) and two landmarks (white circles), showing the pure local graph (left) and the summarized version, where the summarized map is over the set of landmarks. The summarized map for this robot is the factor, colored in blue, connecting the two landmarks. To indicate that the summarized map is over a linearization point in the local reference frame, the variables are colored to match the poses of the original robot.

4.3 Communicating Summarized Maps

With the communication model for DDF-SAM 1.0, a specialized version of the general procedure (see Algorithm 5) I introduce *constrained factor graphs* as a means to fuse summarized maps linearized in separate and unknown reference frames. The complication introduced by local reference frames on separate robots is that the linearization points chosen at the time of summarization will be different. Because the linear densities computed for each system are over an *update* to the existing solution, it is not possible to naively combine linear factors and perform optimization. To solve this problem, I contribute constrained factor graphs to simultaneously solve for a consistent solution for the shared variables, as well as the relative reference frames between robots.

The message passing algorithm for DDF-SAM 1.0 is largely the same as the reference version, with a specialized form of the *addSummarizedMap* operation, which is detailed in Algorithm 10. In this case, note that the system to be updated with the incoming summarized map $\boldsymbol{m}_j^b$ is a neighborhood system $\left(\Phi^{N_r'}, \Theta^{N_r'}\right)$, which can be fully constructed at any given time given the summarized map cache $\mathcal{M}^r$ and initializations for variables, such that the neighborhood map system is strictly the

Algorithm 10 Adding a single summarized map $\boldsymbol{m}_j^b$ from a robot b, summarized at time j to the neighborhood system for robot r.

```
function addSummarizedMap(m_j^b, Φ^{N'_r}, Θ^{N'_r}, M^r):
    find any previously observed m_k^b ∈ M^r
    if timestamp j > k or |M^r| < K
        replace any previous m_k^b in M^r with m_j^b
        extract set of transforms T^r from Θ^{N'_r}
        extract pure local variables Θ^r from Θ^{N'_r}
        (Θ^{N'_r}, Φ^{N'_r}) ← neighborhoodSystem(M^r, Θ^r, T^r)
        Θ^{N'_r} ← optimizeConstrained(Θ^{N'_r}, Φ^{N'_r})
    return (Φ^{N'_r}, Θ^{N'_r}, M^r)
```

fusion of summarized maps.

DDF-SAM 1.0 constructs neighborhood graphs differently than in the local nonlinear system, as the linearized factors comprising summarized maps constrain these maps to the original local frame of reference. As none of the robots know the neighborhood reference frame or the relative reference frames of neighboring robots, I formulate a constrained optimization problem to simultaneously update landmarks in their local frame of reference and in a consistent neighborhood landmark map, as well as solve for the relative reference frames.

Combining these maps requires the neighborhood graph to maintain the separation between the *local side* and the *neighborhood side* of the system, in which I explicitly represent the coordinate frames and the relationships between shared variables observed by separate robots. Figure 14 shows a small three-robot example scenario, in which each robot contributes a factor over three common landmarks, with the construction of a constrained factor graph mapping local- and neighborhood-side landmarks together via a ternary frame of reference constraint. For a neighborhood map built on robot r and given a correspondence (θ_i^r, θ_i^a) between a shared variable in its local reference frame θ_i^a and its counterpart θ_i^r in the neighborhood frame, I

Algorithm 11 Constructing a neighborhood system $\left(\Theta^{N'_r}, \Phi^{N'_r}\right)$ for a robot r with known global correspondences, including initialization of reference frame transforms using matching against the local variables.

`function` $neighborhoodSystem\left(\mathcal{M}^r, \Theta^r, T^r\right)$:

$\Phi^{N'_r} \leftarrow \emptyset$ ▷ `Reconstruct from cache`
$\Theta^{N'_r} \leftarrow \emptyset$
`for each` $\boldsymbol{m}_k^a \in \mathcal{M}^r$
  `if` $a = r$ ▷ `Local summarized map`
    $T_a^r \leftarrow \mathcal{I}$
    `add prior factor` $\phi\left(T_a^r | T_a^r = \mathcal{I}\right)$ `to` $\Phi^{N'_r}$
  `if a transform` T_a^r `exists in` T^r
    `extract` T_a^r `from` T^r
  `else` ▷ `compute new transform`
    $T_a^r \leftarrow computeTransform\left(\Theta^r, \Theta_s^a\right)$
  `add transform` T_a^r `to variables` $\Theta^{N'_r}$
  `add` $factors\left(\boldsymbol{m}_k^a\right)$ to $\Phi^{N'_r}$
  `for each` $\theta_j^a \in variables\left(\boldsymbol{m}_k^a\right)$
    `add variable` θ_j^a `to` $\Theta^{N'_r}$ `in` a`'s reference frame`
    $\theta_j \leftarrow T_a^r \circ \theta_j^a$
    `add transformed variable` θ_j `to` $\Theta^{N'_r}$
    `add reference frame constraint` $c_j^a\left(\theta_j, T_a^r, \theta_j^a\right)$ to $\Phi^{N'_r}$
`return` $\left(\Theta^{N'_r}, \Phi^{N'_r}\right)$

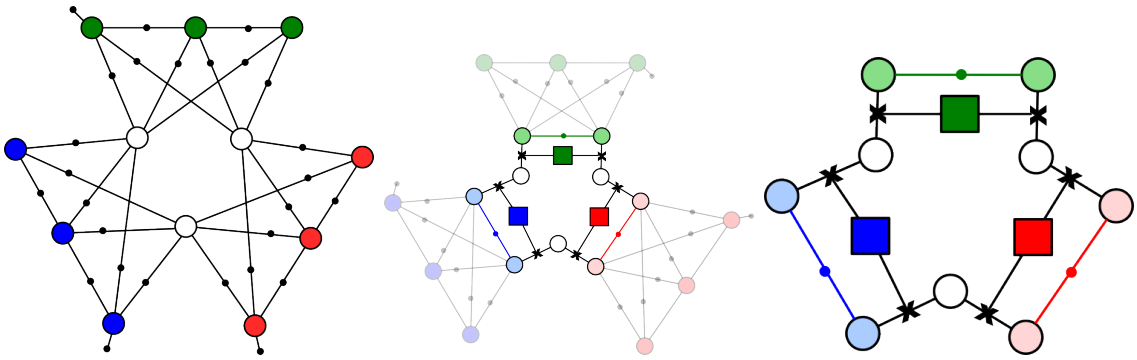


Figure 14: A three-robot scenario case illustrating the construction of a neighborhood graph (right) from summarized maps and the relationship to a naive centralized map (left). In this case, all of the robots have had their local maps summarized to a factor acting on only the landmarks in their local reference frames, as in Figure 13. The center image shows the corresponding structure of a constrained factor graph, with the summarized maps superimposed over the local graphs, and the actual constrained factor graph on the right. In the constrained factor graph, hard constraints are denoted by crosses and reference frame transform variables are denoted by square variables. Landmarks with light coloring are the local copies of a global landmark.

impose an equality constraint

$$T_a^r \oplus \theta_i^a = \theta_i^r \tag{14}$$

between them. Each frame of reference variable T_a^r denotes the rigid-body reference frame transformation mapping shared variables in the neighborhood frame to counterparts in the local frame.

In terms of the size of the resulting optimization problem, each shared variable will be duplicated across separate reference frames, and there will be a ternary frame of reference constraint for each shared variable in each summarized map present. In addition, the frame of reference variables will now be added to the graph, which require initialization, as in the $computeTransform\,()$ step of Algorithm 11. Algorithm 12 describes a simple technique for initialization of transforms that is suitable for shared variables consisting of landmarks. Note that when including factors from the summarized map $\boldsymbol{m}_k^r$ of the local robot r, we assume that the local reference frame transform T_r^r is the identity transform $\mathcal{I}$ and add a prior to ensure that this relationship is maintained during optimization. Adding a prior on at least one of the frame of

Algorithm 12 Computes a rigid-body reference frame transformation between an incoming set of shared variables Θ_s^a from a neighboring robot a and the local variables Θ^r for the local robot r. This approach computes the least-squares solution of a rigid transform between constellations of landmarks ($\theta_i \in \mathbb{R}^n$) with known data associations. In this case, it is assumed there exists a one-one mapping between variables in Θ^r and Θ_s^a, and any non-matching variables are ignored.

```
function computeTransform(Θ^r, Θ_s^a)
    c^r ← centroid(Θ^r)
    c^a ← centroid(Θ_s^a)
    H ← zeros(n, n)
    for each pair (θ_i^r ∈ Θ^r, θ_i^a ∈ Θ^a)
        θ_i^r ← θ_i^r − c^r
        θ_i^a ← θ_i^a − c^a
        H ← H + θ_i^r (θ_i^a)^T
    (U, S, V) ← svd(H)
    R_a^r ← VU^T
    t_a^r ← c^a − R_a^r c^r
    return T_a^r
```

reference transforms is necessary to ensure that the neighborhood graph $\Phi^{N'_r}$ is fully constrained.

4.3.1 Constrained Factor Graphs

I present CFGs as a novel extension of a factor graph as it augments a probabilistic graphical model with deterministic relationships. The hard constraints, which implement the frame of reference constraints (Equation 14), allow for operations such as assignment to be expressed in a graphical manner, as the constraints maintain the separability requirements for graphical models.

The implementation of hard constraints in the underlying nonlinear least squares optimization problem involves only the application of existing techniques [93] for incorporating equality constraints into a least squares optimization problems. I extend the nonlinear least squares problem of Equation 1 to incorporate a set of p equality constraints $c_i\left(\Theta^{N'_r}\right)$ to form a constrained nonlinear least squares problem (Equation

15). These constraint functions are exactly the frame of reference constraints of Equation 14. For convenience, I combine the constraint functions into a single constraint function $g\left(\Theta^{N'_r}\right) \triangleq \left[c_1\left(\Theta_1\right), \ldots, c_p\left(\Theta_p\right)\right]^T$.

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \ \frac{1}{2}\left\|h\left(\Theta^{N'_r}\right) - Z\right\|_\Sigma^2 \tag{15}$$
$$subject\ to\ c_j^i\left(\Theta\right) = 0\ \forall i \in [1 \ldots p]$$

This optimization procedure has been shown [93] to be equivalent to optimizing a quadratic merit function

$$\varphi(X) = \frac{1}{2}\left\|h\left(\Theta\right) - Z\right\|_\Sigma^2 + \mu\frac{1}{2}\left\|g\left(\Theta\right)\right\|^2 \tag{16}$$

with a sufficiently high μ parameter. At each linearization stage, I approximate the system using a first order Taylor expansion as in the unconstrained case (Equation 3), where G is the Jacobian of $g\left(\Theta\right)$ evaluated at Θ. I solve the linearized system with direct elimination of the constrained variables.

$$\delta^* = argmin_\delta \frac{1}{2}\left\|A\delta - b\right\|_\Sigma^2 \tag{17}$$
$$subject\ to\ G\delta + g(\Theta) = 0$$

To solve the linear subproblems, I use a hybrid elimination procedure eliminating variables with only probabilistic factors using the Householder reflections exactly as in the probabilistic case, and use Gram-Schmidt orthogonalization to eliminate variables with hard constraints.

4.4 Evaluation

To validate the approach, I performed experiments using both real-world robot tests and the simulated datasets described in Section 3.4.6, demonstrating not only the performance predicted by the theory, but also adaptability to real-world scenarios and hardware. The evaluation scenarios include both the Manhattan-world and large-scale

simulated datasets, described in Section 3.4.6, as well as real robot scenarios using both robots with direct teleoperation as well as fully autonomous robots performing exploration.

The primary evaluation metrics for these scenarios are those described in Sections 3.4.3, 3.4.4 and 3.4.5, measuring online operation with timing performance, scalability with communication bandwidth, and consistency with estimation error, respectively. For scenarios without ground truth information, or those too small to be statistically meaningful, we use qualitative analysis of map quality.

As an implementation detail of the experimental setup, it is possible to use differing types of local mapping solvers for the local update phase of DDF-SAM - a batch solver, such as Levenberg-Marquardt, or a standard incremental solver, such as iSAM 2.0 [85]. In the evaluations, I use iSAM 2.0 as the local solver for the system as it would be the most typical choice for a single-robot online SAM solver. Map summarization and neighborhood map fusion remain identical to the approach described in this chapter.

4.4.1 Real-Robot Scenarios

The following two sections detail the real-world robot experiments performed, which were specific to the DDF-SAM 1.0, with two datasets on different robot platforms, which convincingly show that it is possible to apply DDF-SAM across sensing modalities and map representations. The first dataset is the Freiburg dataset, detailed in Section 3.4.6.3. The second experiment combined DDF-SAM with fully autonomous exploration with multiple landmark types in collaboration with the University of Pennsylvania, which I detail below.

4.4.1.1 UPenn Robot Experiments

We demonstrated [73] that DDF-SAM can work in a fully autonomous decentralized exploration system through experiments conducted at UPenn, using their custom

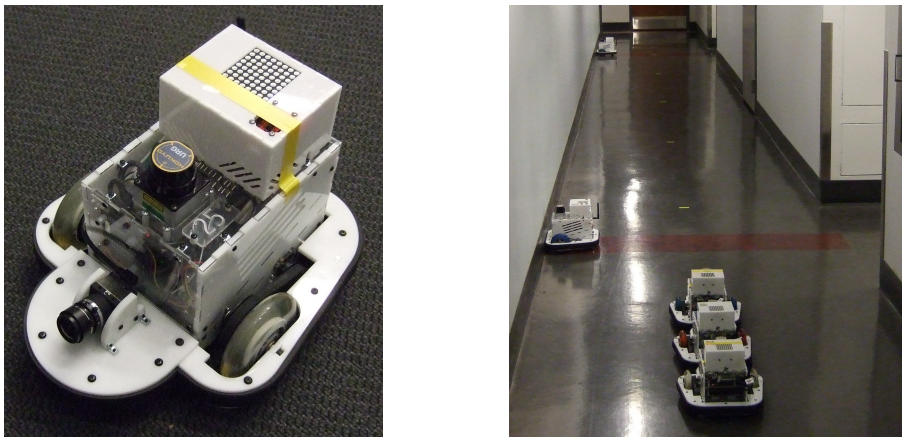


Figure 15: Scarab robots used for the UPenn experiments, with a closeup (left) of an individual robot, and the experimental setup for autonomous exploration at the start of the scenario. Each robot is equipped with a Hokuyo laser scanner and a forward-facing camera and 802.11s mesh networking hardware. In this scenario, there are three robots (shown in a line at the center of the starting case) exploring autonomously, with additional robots placed through the environment to act as network repeater nodes to allow for communication to a base station. All mapping is performed locally on each robot.

Scarab robots, in which we performed mapping onboard the robots using real-world mesh networking hardware. In this scenario, the robots autonomously explored an office environment while performing mapping in real time. This scenario, shown at its starting point in Figure 15, highlights both online operation, and the ability to use multiple sensing and representation types, as maps in this environment were comprised of both line segments extracted from a Hokuyo scanning laser rangefinder and doors detected using an front-mounted camera. The robots employed a mesh network system for communication, both for between-robot map sharing and coordination, but also for sending map information back to a base station computer for visualization. Note that the base station computer was only used for visualization; all of the decentralized mapping was performed onboard the robots at real-time.

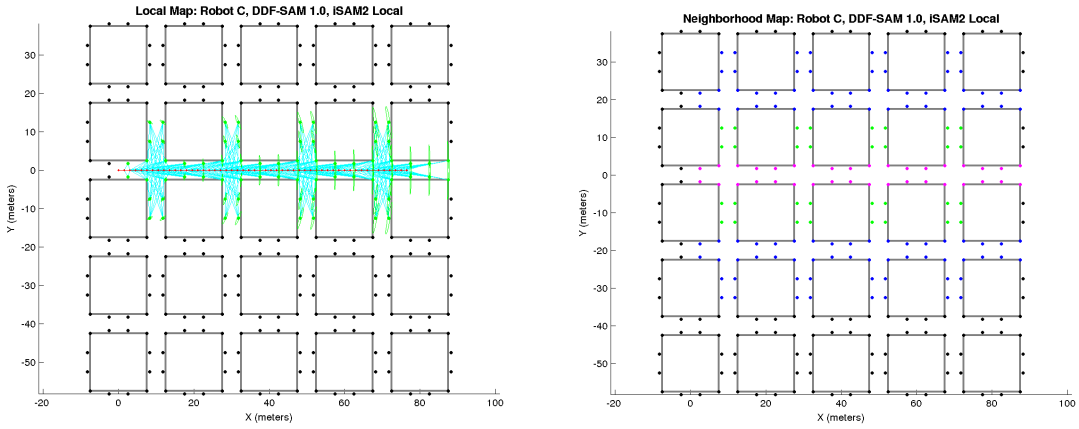


Figure 16: Manhattan-world results for DDF-SAM 1.0, showing the local (left) and neighborhood maps for a single robot with two neighbors. The neighborhood map shown is color coded by variable membership, grouped by pure local, overlapping, and pure neighborhood variables, colored magenta, green and blue, respectively.

4.5 Results

This section presents the results of the experiments described in the previous section, which serve to validate the core claims that DDF-SAM is online, scalable and consistent. The remainder of this section will present the results of experiments I performed using DDF-SAM 1.0, starting with the smaller Manhattan-world scenario as an illustrative example, then the real-world experimental scenarios, followed by the large scale simulation for aggregate statistics.

4.5.1 Manhattan-world Simulation

The Manhattan-world simulation results, shown in Figure 16, show that the sensor horizon of the target robot is extended beyond the local map. This example serves as a pedagogical example for illustrating the results of the algorithm in order to better understand how the real-world datasets work.

Because DDF-SAM 1.0 maintains two maps, Figure 16 illustrates the map solution of both the pure local mapper as well as neighborhood map constructed from summarized maps fused from two neighboring robots. The local map is, as expected, identical to a pure local SAM system, generating a map solution over the locally

observed landmarks and robot trajectory. The neighborhood map solution, over only the landmarks, show how DDF-SAM extends the sensor horizon of an individual robot by grouping variables by the their relationship to the current map. The neighborhood map shows three groups of variables

- Pure local variables (rendered as magenta): Variables observed by only the local robot.
- Overlapping variables (rendered in green): Variables observed by both the local robot and at least one neighbor.
- Pure neighborhood variables (rendered as blue): Variables only present through received neighborhood maps, and not present in the local map. These form the extended sensor horizon of the robot.

As can be seen from the rendered neighborhood map result for a single robot in the Manhattan-world scenario in Figure 16, the extended sensor horizon covers a much larger area then the local map, which validates the claim that DDF-SAM enables robots to aid the map solutions of neighbors.

4.5.2 Real-Robot Scenarios

To evaluate DDF-SAM 1.0, I ran two real-world experiments from datasets in from Freiburg and UPenn, that validate DDF-SAM, highlighting several useful characteristics

- Online: Demonstrated primarily by the UPenn dataset as mapping occurred onboard the robots.
- Consistent: While neither dataset has ground truth, it is possible to examine the map consistency qualitatively from renders of map solutions. In addition, I use the centralized solution as ground truth in the Freiburg dataset case.

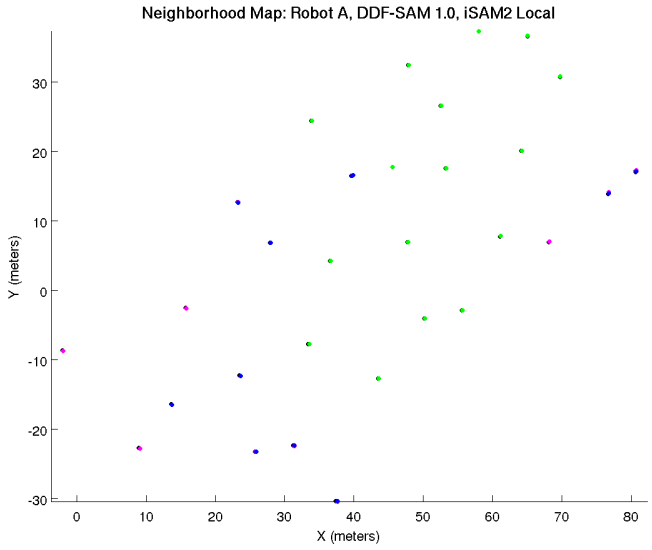


Figure 17: Neighborhood landmark map rendering from robot A in the Freiburg dataset run 1. The landmarks are color-coded exactly as in Figure 16. Black landmarks denote the ground truth solution. Note that the map is in the local reference frame of the robot.

- Extended horizon: Both datasets demonstrate map fusion, resulting in an extended sensor horizon.

- Variety of sensing modalities: Between the two datasets, there are three sensing modalities, strongly proving the claim that DDF-SAM is flexible.

However, these experiments are not able to provide an evaluation of scalability, as the robot teams in each case are no larger than three robots - I address this shortcoming through the use of large scale simulations in the next section.

4.5.2.1 Freiburg Dataset

Figure 17 shows neighborhood map generated through DDF-SAM 1.0 for one of the robots in the dataset, in which it is clear that the additional information received from neighboring robots has extended the sensor horizon beyond the locally observed landmarks. In both runs for this dataset, the robots largely cover much of the same area, so a large portion of the landmarks are shared between robots. Note that while there is visibly observable error in this system, it is still at a manageable level.

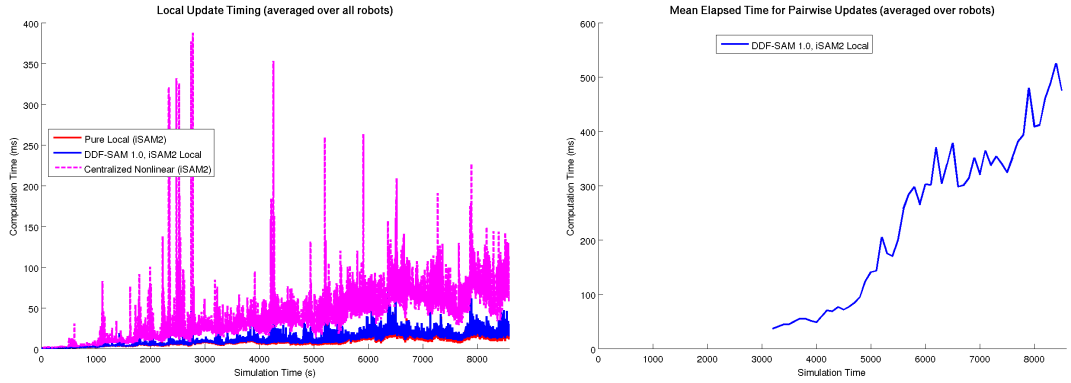


Figure 18: Update timing for update operations in DDF-SAM 1.0 on Freiburg dataset run 2, showing the time for local updates (left) and for neighborhood updates. Note that neighborhood updates did not start until the robots began communication when there were enough common landmarks.

To evaluate the update performance for the system, I measured time for adding local measurements as well as for fusing an incoming summarized map, as shown in Figure 18. As comparisons, I also show the update timing results for both a pure local single-robot iSAM 2.0 solution and a centralized incremental solution across all robots. In this case, the centralized solver is predictably slower due to the larger system being solved. The difference between the pure local solver and the DDF-SAM 1.0 solver is small, with DDF-SAM 1.0 taking approximately 5 ms longer per update than the pure local solution. This aligns with predictions, as both are solving the same problem, though the DDF-SAM 1.0 solver has some additional overhead to account for tracking variables. The neighborhood map update timing, being a full batch optimization, increases quickly with time, which is to be expected. Note that the results start later in the dataset as the robots needed to encounter sufficient overlapping landmarks before communication could begin.

I used the batch ground truth solution to measure the error of the landmark estimation over time in the Freiburg dataset, as illustrated in Figure 19. Note that in this case the centralized solution has large error spikes in the beginning, which is before there were many variables in common, which predictably yields large errors

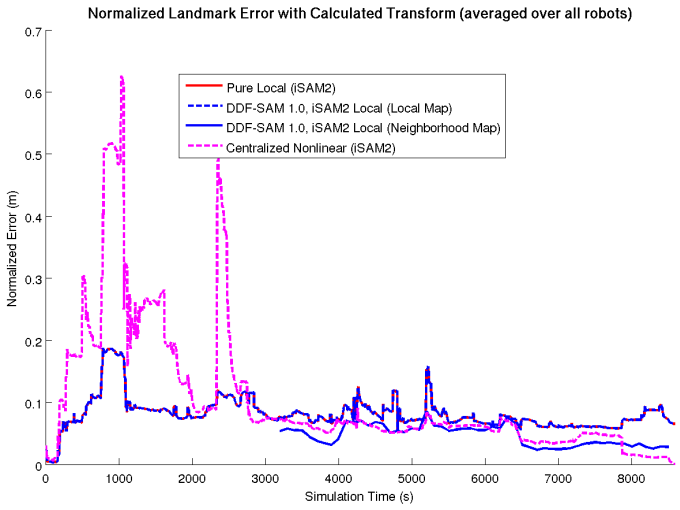


Figure 19: Landmark error constellation error for DDF-SAM 1.0, using calculated transforms to align to the ground truth solution, with comparisons to a pure local and a centralized iSAM 2.0 solver. Note that he pure local and DDF-SAM 1.0 local map solutions are identical.

comparing the to the ground truth solution. Communication only actually starts at approximately timestep 3200 due to the need for overlapping variables. As a positive result, as the solutions converge after communication begins, the neighborhood solution for the system has lower error than the local solution, and converges towards the batch optimization ground truth near the end of the dataset.

In Figure 20, I show statistics for the summarization phase of DDF-SAM - both the time necessary to compute the summarized map and the communication cost measured as the size for a given summarized map. The computation time for summarization appears to be approximately linear with time, which is to be expected the local graph for each robot is increasing in size with each timestep due to additional poses. To evaluate communication cost, I used an approximation of the communication cost for a centralized system in which each robot sends raw measurements to a central location, as accumulated between communication timesteps. The spike in centralized cost at the start is due to the long delay (1000 timesteps) before robots were allowed to begin communicating. In this dataset, because there are a relatively

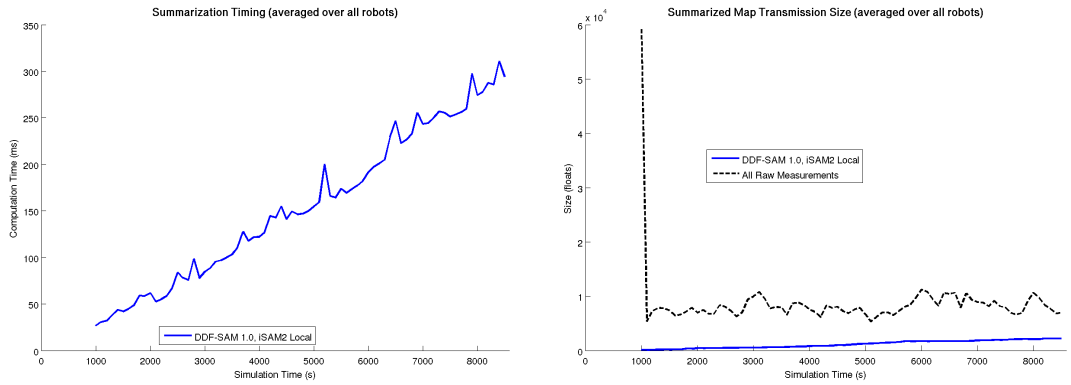


Figure 20: Summarization statistics for Freiburg run 2, showing the time to compute the summarization (left) and the summarized map transmission size. The summarized map transmission size also includes an approximation of a centralized communication cost, which accumulates measurements between communication timesteps.

small number of landmarks in the environment (on the order of 30), the growth rate is relatively flat in comparison to the time, since the trajectories are quite loopy.

4.5.2.2 UPenn Experiment

The UPenn experiment [72] validates the claim that DDF-SAM is both efficient enough for online decentralized mapping using only on-board robot hardware, and generalizable enough to work with a variety of sensing modalities and environmental representations. A plot of the neighborhood map generated by three robots exploring the environment appears in Figure 21 with a ground-truth floorplan of the office environment for reference. During exploration and mapping, we measured the compute resources required by the decentralized mapping process, and the DDF-SAM process consumed a small fraction of the available processing power available on the already computationally limited platforms, demonstrating the claim that DDF-SAM can operate in online.

The representation used, combining expandable line segments [57, 59] to model walls and oriented door objects, validated the claim that DDF-SAM is flexible enough for a variety of sensing modalities, as well as showing how a good choice of shared variables can yield reliable performance. Because the line segment features, which

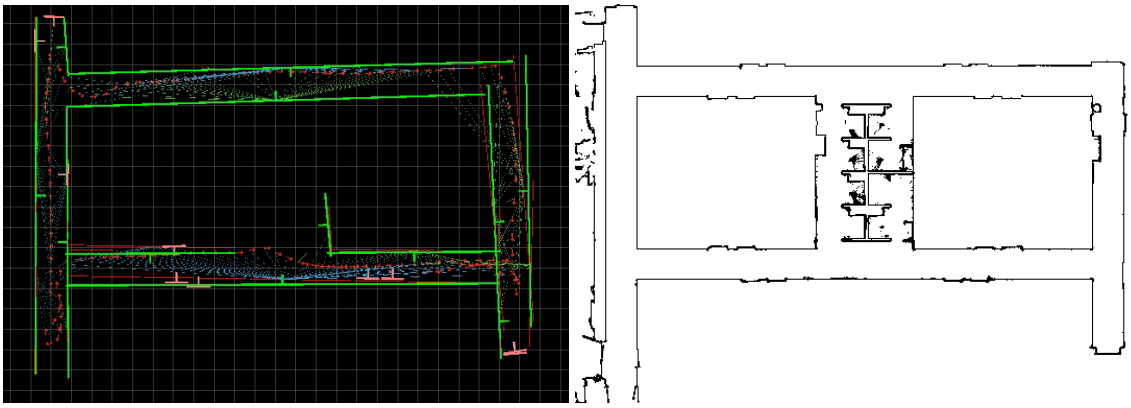


Figure 21: Neighborhood map result from three robots autonomously exploring an office environment in the UPenn experiment, showing the neighborhood map (left) composed of line-segment walls and doors, and a ground-truth floorplan created from raw laser scan data. In the neighborhood map, green lines indicate neighborhood-optimized line segments, and pink lines indicate doors. Trajectories and measurement lines are shown under the neighborhood-optimized features.

comprised a majority of the structure in the map, can expand as the robots explore, they can represent the environment sparsely, using on the order of 30 variables to the model the environment, rather than the hundreds or thousands that a denser SLAM approach like visual SLAM might. This sparsity of representation yields good performance within the mapping framework.

4.5.3 Large Scale Simulation

The large scale simulation convincingly shows the following claims are substantiated:

- Online: Local updates are no worse than the local mapping solution used.
- Scalable: I demonstrate that fusion neighborhood bounding limits both computational requirements and communication bandwidth.
- Consistent: I provide error metrics for the map solutions as compared to both centralized solving and local-only mapping.

However, these results below also show:

- Online: batch neighborhood updates and summarization are expensive.

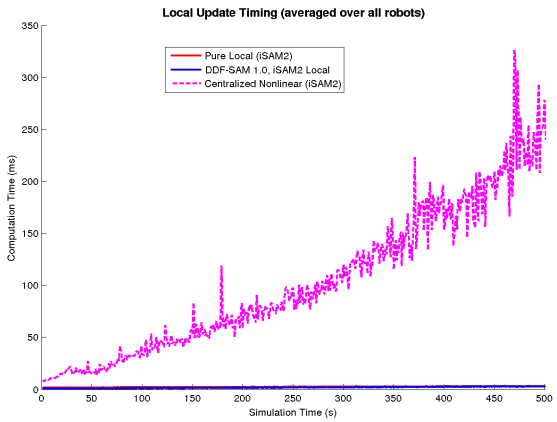


Figure 22: Local update performance for DDF-SAM 1.0 using an iSAM 2.0 local solver, compared to a centralized solution and a pure local solution.

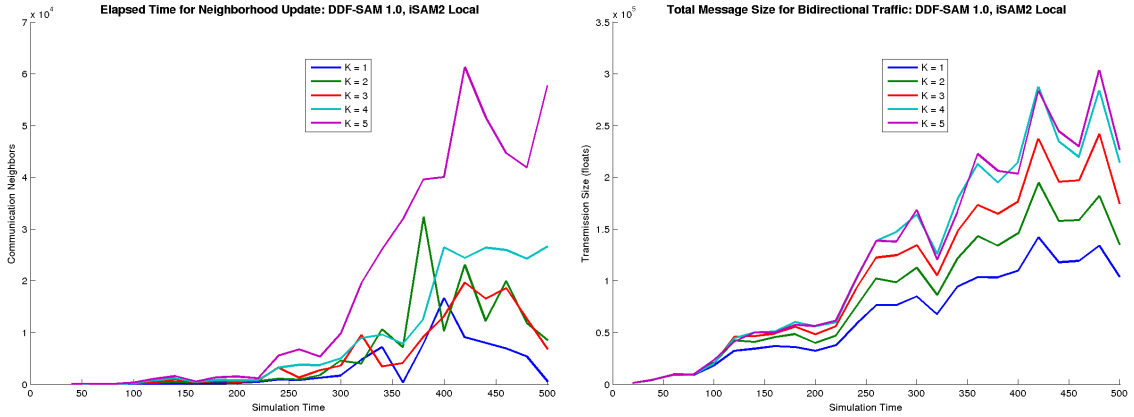


Figure 23: Comparison of the effect of bounding the number of fusion neighbors, showing effect on compute timing for adding summarized maps (left) and the traffic through a single node.

- Scalability: limited due to the quadratic growth in transmission size of summarized maps.

These shortcomings provide the basis for the extensions of DDF-SAM in later chapters.

Figure 22 shows computational performance for local updates, as compared to a centralized and pure local system, and validates the claim that local mapping will be substantially faster than a centralized approach, and that the DDF-SAM local mapping solution is no slower than a single-robot mapping system.

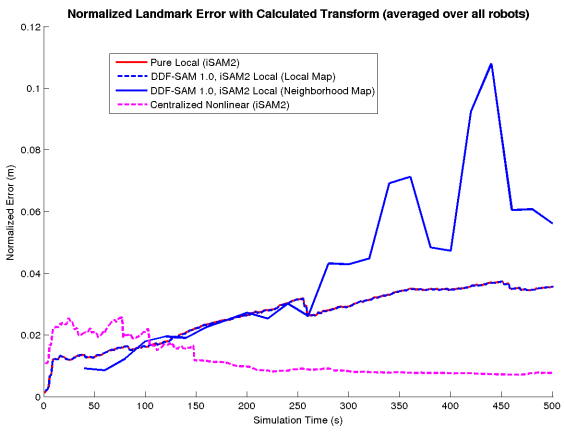


Figure 24: Landmark constellation error, normalized to error (in meters) per landmark, showing error for the DDF-SAM 1.0 local and neighborhood solution, as well as a pure local solution. Note that the DDF-SAM 1.0 local map error and the pure local reference solver have identical curves.

I validate the claim that bounding the number of fusion neighbors for a given robot to a hard maximum K act as a bound that can throttle performance requirements for both computation and communication, as plotted in Figure 23. These figures compare different values of K for performance in neighborhood updates and communication bandwidth, and in each case, higher values of K require more computation during updates as well as greater message bandwidth. Therefore, by adjusting K it is possible to adapt performance of the system to meet the capabilities of a particular robot platform.

As a metric of solution consistency, I measured the normalized landmark constellation error of the solution for the landmarks, shown in Figure 24, which demonstrates error in the DDF-SAM neighborhood map is acceptable for mapping purposes. Note that in this case, the oscillations in error for the neighborhood map solution can be attributed to variations in frame of reference solutions. While the solution error is higher than both the centralized and pure local solutions, this can be expected in part because nonlinear optimization in the CFG for the neighborhood map solution is not able to relinearize factors from summarized maps.

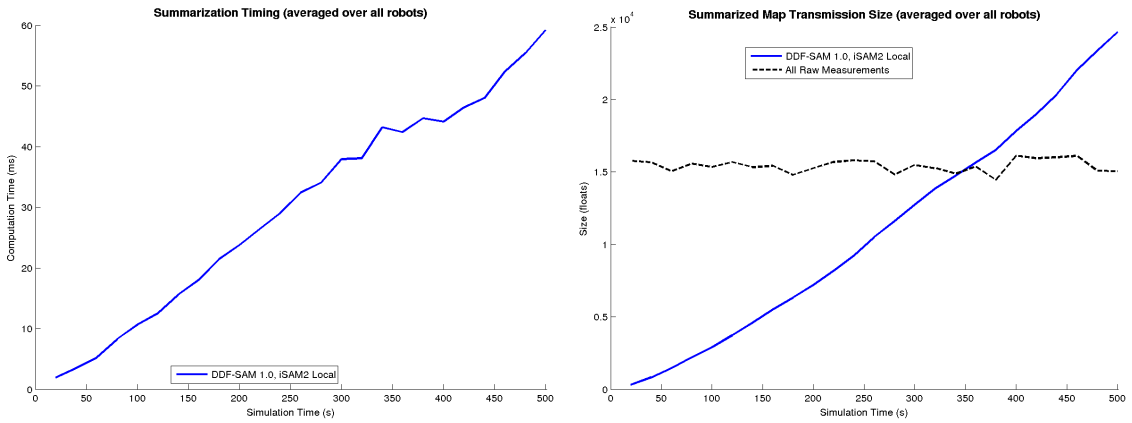


Figure 25: Summarization timing (left) and summarized map transmission size for DDF-SAM 1.0. As a comparison for summarized map transmission size, the plot also shows a transmission size for sending all nonlinear measurements since the last communication interval, denoted as "All raw measurements."

To evaluate the summarization process in terms of both online operation and scalability, I measured performance timings for creating a summarized map and the map transmission size, which highlight weaknesses with the current DDF-SAM approach. Using the current summarization process, which relies on performing direct partial elimination on the entire system, the computational cost of summarization message size grow substantially over time. The growth in the map transmission size is also problematic, as the cost of sending a single summarized map surpasses sending all raw nonlinear measurements since the last communication interval. This growth rate in the size of summarized map forms the motivation for the use of approximate summarization techniques, detailed in Chapter 7.

4.6 Discussion

DDF-SAM 1.0 has proved to be an effective means implementing a decentralized SAM architecture while ensuring consistency through a strict separation between local and neighborhood maps, and yielding an approach that is demonstrably online, scalable, consistent and flexible enough for a variety of sensing modalities.

As described in this chapter, DDF-SAM 1.0 has two key shortcomings: 1) an overly

conservative approach for avoiding information double-counting, and 2) reliance on a batch marginalization approach for map summarization. The method employed for avoiding information double-counting enforced a strict separation between a robot's local map and the neighborhood map comprised of data from neighboring robots - an approach that avoids double-counting, but results in each robot maintaining two separate, but incomplete maps of its environment. Furthermore, the batch summarization technique used to marginalize non-shared variables from the local map performs a computationally expensive and separate factorization of the entire local system, which increases in complexity over time and limits scalability.

Chapter 5

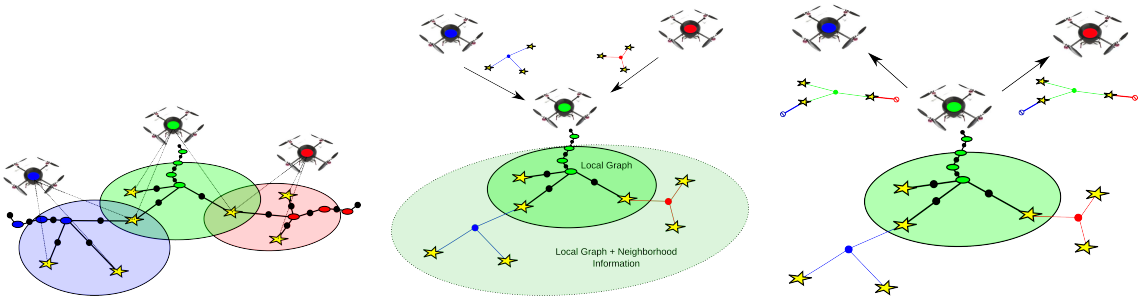
DDF-SAM 2.0: INCREMENTAL DECENTRALIZED SMOOTHING AND MAPPING

To address the shortcomings present in the DDF-SAM 1.0 approach described in Chapter 4, I introduce in this chapter DDF-SAM 2.0 [33, 32], which contributes the *augmented local system* as a replacement for the separate local and neighborhood maps maintained in DDF-SAM 1.0 to provide a single, consistent map on each robot blending local and neighborhood information. Figure 26 illustrates the overall approach in a three-robot scenario. In order to maintain consistency, I introduce the *antifactor* as a tool to avoid double-counting of neighborhood information by downdating estimates.

DDF-SAM 2.0 takes a structurally different approach to the DDF-SAM problem than the DDF-SAM 1.0 approach, while still providing contributions in the following areas:

- Online: DDF-SAM 2.0 centers around a single incremental solver, extending iSAM [86, 85], which improves upon DDF-SAM 1.0 by avoiding batch nonlinear updates for faster performance.
- Scalable: DDF-SAM 2.0 maintains the same communication model as the general DDF-SAM.
- Consistent: DDF-SAM 2.0 combines local and neighborhood maps to create a single, internally consistent map with both local and neighborhood information.

As in all of the DDF-SAM variations, I approach the problem as a generalized graphical inference problem, so that it will remain sufficiently flexible for a variety of sensing



(a) Three-robot Scenario (b) Adding Neighborhood Information (c) Summarization with antifactors

Figure 26: An example 3-robot scenario (left), with the augmented local system of the green robot integrating summarized map information from neighboring robots (center), and sharing summarized map information with antifactors (denoted with crossed circles). Note the expanded region of coverage through the addition of summarized neighborhood information.

modalities and map representations. This representation and corresponding algorithm enables each robot to maintain a consistent SLAM solution with an extended sensor horizon provided by neighboring robots. The full DDF-SAM 2.0 formulation appears in Algorithm 13.

The key distinctions between DDF-SAM 2.0 and DDF-SAM 1.0 can be summarized as follows

- Rather than avoiding information double-counting by isolating local and neighborhood maps and only summarizing a pure local map, DDF-SAM 2.0 corrects for the double-counted information at the summarization stage. This allows each robot to maintain a single map with all available information.

- DDF-SAM 2.0 uses an incremental solver formulation to minimize the cost of adding new measurements, rather than a performing nonlinear optimization on a full local or neighborhood system as in DDF-SAM 1.0.

As a pedagogical simplification, I present this algorithm as in [33, 32] using a single-linearization incremental approach that assumes all measurements are linearized immediately and linearization points are shared between robots in a global reference

Algorithm 13 DDF-SAM 2.0 Algorithm.

initialize: for each robot r

$\Theta^{N'_r} \leftarrow \{x^r_{init}\}$ ▷ initialize neighborhood variables
$\mathcal{B}^{N'_r} \leftarrow \emptyset$ ▷ Bayes tree (neighborhood)
$\mathcal{M}^r \leftarrow \emptyset$ ▷ local summarized map cache

for each robot r at each discrete timestep k

$\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}\right) \leftarrow processLocalMeasurements\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, k\right)$
$\Theta^r_s \leftarrow chooseSharedVariables\left(\Theta^r\right)$
if k is a summarization interval and $|\Theta^r_s| > d_{min}$
$\boldsymbol{m}^r_k \leftarrow summarize\left(\mathcal{B}^{N_r}, \Theta^r_s, k, \mathcal{M}^r\right)$
add $\boldsymbol{m}^r_k$ to cache $\mathcal{M}^r$ and any replace older $\boldsymbol{m}^r_{k-1}$
$\left(\mathcal{M}^r, \mathcal{B}^{N'_r}, \Theta^{N'_r}\right) \leftarrow communicate\left(\mathcal{M}^r, \mathcal{B}^{N'_r}, \Theta^{N'_r}\right)$

frame. I relax this assumption in the full nonlinear version of DDF-SAM 2.0 detailed in Chapter 6.

5.1 Processing Local Measurements

For DDF-SAM 2.0, I use an existing incremental update procedure [86] in order to ensure updates sufficiently fast for online operation by re-solving only the necessary portions of the full graphical model. As has been shown in previous work incremental SAM (see Section 2.5.3), performing incremental updates will typically always be faster than performing full batch optimization, as the worst case, where the full linear problem needs to be re-solved, is the same as performing a single iteration Gauss-Newton (detailed in Algorithm 9).

The local processing stage, presented as pseudocode in Algorithm 14, of DDF-SAM 2.0 uses a simple iSAM approach [86] in which the state for the map is maintained by storing and updating a sparse, pre-factorized linear system by a Bayes tree $\mathcal{B}^{N'_r}$ over both local and neighborhood measurements, as well as a full set of variables $\Theta^{N'_r}$. Note that we indicate a system covering variables from fusion neighbors as well as the local robot r through the set N'_r. As formulated in Algorithm 14, the local

Algorithm 14 The local phase of a DDF-SAM 2.0 solver performs incremental updates to a Bayes tree $\mathcal{B}^{N'_r}$. Note that this approach linearizes local factors once, rather than maintaining a full graph $\Phi^{N'_r}$.

```
function processLocalMeasurements(B^{N'_r}, Θ^{N'_r}, k):
    collect measurements Z_k^r from local sensors
    create local factors Φ_k^r from Z_k^r
    for each new factor φ_j^r ∈ Φ_k^r
        for each variable θ_i ∈ variables(φ_j^r)
            if θ_i has not been observed
                z_j^r ← measurement(θ_i^r)
                θ_i ← initializeVariable(Θ^{N'_r}, z_j)
                add θ_i to Θ^{N'_r}
    Φ̃_k ← linearize(Φ_k, Θ^{N'_r})
    Θ_k ← variables(Φ̃_k)
    B^{N'_r} ← linearUpdateIncrementalSolver(Φ̃_k, B^{N'_r})
    return (B^{N'_r}, Θ^{N'_r})
```

update process is largely identical to the generic DDF-SAM approach (see Algorithm 2), with two key distinctions

- The system state for a robot r is represented as a neighborhood Bayes tree $\mathcal{B}^{N'_r}$ and a linearization point $\Theta^{N'_r}$, rather than maintaining a pure local graph Φ^r and linearization point Θ^r.
- The update procedure uses an incremental update the Bayes tree with a set of new factors.

The use of a neighborhood Bayes tree $\mathcal{B}^{N'_r}$ and linearization point $\Theta^{N'_r}$ are significant not only due to the use of an incremental update procedure, but also due to the inclusion of neighborhood information in a partially factorized linear system. Within a neighborhood factor graph $\Phi^{N'_r}$ containing separate factors for each local measurement and each summarized map from a neighboring robot it is possible to separate local and neighborhood contributions through measurement tagging. However, by maintaining

Algorithm 15 Linear update to an incremental Bayes tree solver.

```
function linearUpdateIncrementalSolver(Φ̃_k, B^{N'_r}) :
    B_orphans ← ∅
    Θ_k ← variables(Φ̃_k)                                  ▷ affected set
    Φ̃_update ← liquefyBayesTree(B^{N'_r}, Θ_k)
    for each affected variable θ_j ∈ Θ_k
        remove clique C(θ_j) and path to root in B^{N'_r}
    add orphaned subtrees of B^{N'_r} to B_orphans
    add new linear factors Φ̃_k to Φ̃_update
    B^{N'_r} ← multifrontalEliminate(Φ̃_update)
    insert orphans B_orphans into B^{N'_r}
    return B^{N'_r}
```

a pre-factorized version of $p\left(\Theta^{N'_r}\right) \propto \left(\mathcal{B}^{N'_r}, \Theta^{N'_r}\right)$, these contributions cannot be easily separated, the implications of which I will discuss during summarization in Section 5.2.

The incremental update step $linearUpdateIncrementalSolver\left(\tilde{\Phi}_k, \mathcal{B}^{N'_r}\right)$, detailed in Algorithm 15, that occurs in local measurement processing performs a linear version of the full iSAM algorithm, in which given an existing Bayes tree $\mathcal{B}^{N'_r}$ and a new set of linear factors $\tilde{\Phi}_k$, I re-eliminate only the relevant portions of the Bayes tree. This algorithm starts by reinterpreting the affected cliques of the Bayes tree as a set of linear factors $\tilde{\Phi}_{update}$ through a process known as liquefying a Bayes tree, detailed in Algorithm 16, which are combined with the new factors $\tilde{\Phi}_k$, which contains all of the portions of the Bayes tree relevant to the incoming factors. The algorithm proceeds by removing the affected cliques within the tree, leaving a set of smaller orphaned subtrees that are the parts of the system not affected the incoming factors. I assemble the final updated Bayes tree from the result of multifrontal elimination over $\tilde{\Phi}_{update}$ combined with the orphaned subtrees.

Algorithm 16 Liquefying a Bayes tree $\mathcal{B}$ into a factor graph over a set of affected variables Θ_k.

```
function liquefyBayesTree(B, Θ_k) :
    Φ̃ ← ∅                                  ▷ Start with empty graph
    for each affected variable θ_i ∈ Θ_k
        find clique C(θ_i)
        for each clique C_j in path from C(θ_i) to root of B
            φ̃_j ← interpretAsFactor(C_j)
            add φ̃_j to Φ̃ if not already added
    return Φ̃
```

Liquefaction of a Bayes tree, described in Algorithm 16, is a process of reinterpreting both the affected cliques $\mathcal{C}(\theta_i)$ for each affected variable $\theta_i \in \Theta_k$, but also all of the parent cliques to the root of the tree. This operation forms the basis of not only performing incremental updates, but also computing joint densities over shared sets of variables, which will be used for summarization.

The incremental update procedure described in this section is the simplest variation, and does not employ relinearization or variable reordering, which will become necessary for most practical systems due to fill-in during multifrontal elimination. During evaluation in this chapter, I extend the procedure to include periodic variable reordering, which prevents the Bayes tree $\mathcal{B}^{N'_r}$ from becoming too dense, as will likely happen with a naively chosen variable ordering.

5.2 Summarizing Local Map Information

I contribute a new map summarization approach, described in Algorithm 17, that correctly accounts for double-counted map information maintained within the full Bayes tree $\mathcal{B}^{N'_r}$, ensuring both consistent data fusion for receiving robots, as well as a minimum of additional computation to perform summarization. This map summarization technique exploits the existing factorized system encoded in the Bayes tree,

Algorithm 17 Summarizing over locally observed shared variables directly from a Bayes Tree while accounting for double-counted information with an antifactor correction.

`function` $summarizeFromBayesTree\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Theta^r_s, k, \mathcal{M}^r\right)$:

$\tilde{\Phi}_{full} \leftarrow liquefyBayesTree\left(\mathcal{B}^{N_r}, \Theta^r_s\right)$
$\tilde{\Phi}^r_{summarized} \leftarrow summarizeLinear\left(\tilde{\Phi}_{full}, \Theta^r_s\right)$
`for each` $\boldsymbol{m}^a_j \in \mathcal{M}^r$
    $\tilde{\Phi}^{-a}_j \leftarrow subtractDoubleCounted\left(\boldsymbol{m}^a_j, \Theta^r_s\right)$
    `add antifactors` $\tilde{\Phi}^{-a}_j$ `to` $\tilde{\Phi}^r_{summarized}$
$\tilde{\Phi}^r_{summarized} \leftarrow condenseFactors\left(\tilde{\Phi}^r_{summarized}\right)$
$\boldsymbol{m}^r_k \leftarrow createSummarizedMap\left(\tilde{\Phi}^r_{summarized}, \Theta^r_s, k\right)$
`return` $\boldsymbol{m}^r_k$

---

and performs antifactor down-dating to correct for information that would be double-counted. The resulting summarized map is exact, and adheres to the summarization requirement that summarized maps contain only local information.

At the start of the approach, I liquefy and summarize the Bayes tree $\mathcal{B}^{N'_r}$ to yield a linear density over the locally observed shared variables Θ^r_s, which provides performance benefits during map summarization. By starting with the existing Bayes tree, I skip the the linearization step necessary for starting linearization from a full factor graph (as in Algorithm 3 used throughout DDF-SAM 1.0). In addition, because liquefying a Bayes tree only creates factors for those along the path from each target variable to the root variable, there are sections of the Bayes tree that can be ignored, yielding a smaller linear system $\tilde{\Phi}_{full}$ from which to start linear summarization.

The core contribution of this summarization approach is the correction for double-counted information, in which I add to $\tilde{\Phi}^r_{summarized}$ a series of *antifactors*, which have the effect of subtracting the information contribution from the cache summarized maps. This step ensures that the resulting summarized map $\boldsymbol{m}^r_k$ has an information contribution equivalent to performing summarization on the pure local factor graph

Φ^r. The derivation and details of this technique appear in the following section.

The final step for summarization is to condense the collection of linear factors $\tilde{\Phi}^r_{summarized}$ into a smaller set of linear factors for communication by converting the factors into information form and adding the information matrices directly. This prevents antifactors from adding to the size of the map summarization.

5.2.1 Antifactor Down-dating

I contribute antifactor down-dating as a tool for subtracting information from a fused estimate as a tool for exactly subtracting double-counted information from summarized maps to ensure consistent estimation. This technique is similar to operations performed in a channel filter algorithm [99], in that it corrects explicitly for double-counted information to maintain consistency. The formulation exploits the additive nature of information matrices to allow for subtracting known information contributions from an existing fused density.

A necessary component of this process is the relationship between linear factors in square-root information (Jacobian) form and in information (Hessian) form. For a multivariate linear factor with N variables $\delta_i \in \Delta$, defined as

$$\tilde{\phi}(\Delta) \triangleq \frac{1}{2} \left\| \sum_{i=1}^{N} A_i \delta_i - b \right\|^2, \tag{18}$$

the linear system can be represented in square root information form as (A, b), where the measurement Jacobian $A \triangleq [A_1, ..., A_N]$. Carrying out the multiplication in the norm Information form,

$$\tilde{\phi}(\Delta) \triangleq \frac{1}{2} \left(\Delta^T G \Delta - 2\Delta^T g + f \right) \tag{19}$$

where G is the Hessian of the measurement function and is computed as $G \triangleq A^T A$. Note that the Hessian is actually the information matrix Λ in this formulation.

The key difference exploited to enable down-dating is that, unlike in the Jacobian form (18), factors on the same variables in Hessian form can be combined through

Algorithm 18 Subtracting double-counted information from a summarized map.

`function` $subtractDoubleCounted\left(\boldsymbol{m}_j^a, \Theta_s^r\right)$:

$\Theta_s^a \leftarrow variables\left(\boldsymbol{m}_j^a\right)$
$\tilde{\Phi}_j^a \leftarrow factors\left(\boldsymbol{m}_j^a\right)$
$\Theta^{ar} \leftarrow \Theta_s^a \bigcap \Theta_s^r$ ▷ `determine overlapping variables`
$\tilde{\Phi}^a \leftarrow summarizeLinear\left(\tilde{\Phi}_j^a, \Theta^{ar}\right)$
`return` $\left(\tilde{\Phi}^a\right)^{-1}$ ▷ `negate contribution of` a `on` Θ_s^r

direct addition of Hessian matrices G. In the combination of M linear factors $\tilde{\Phi}$, the final Hessian matrix is computed by

$$G \triangleq \sum_{j=1}^{M} G_j.$$

This additive property is matches well with the intuition that adding new measurements in the form of factors should then add information to the estimate, and has been exploited by information filtering techniques as an approach for reducing the cost of adding information to a system.

However, while it is possible to add information to an estimate, it is possible to subtract information from an estimate as well, using simple negation of the Hessian, a process which I will refer to as creating an antifactor. For any linear factor $\tilde{\phi}\left(\Delta\right)$ in Hessian form, I can construct an antifactor $\tilde{\phi}^{-1}\left(\Delta\right)$ by simple negation of G. To remove the contribution of a specific factor $\tilde{\phi}_{target}$ from a factor graph $\tilde{\Phi}$, it is simply necessary to add the corresponding antifactor $\tilde{\phi}_{target}^{-1}$ to the graph and combine Hessian matrices.

Because the summarized map cache in DDF-SAM contains the exact contribution made by each summarized map to the augmented local system, the process of subtracting double-counted information simply requires constructing an antifactor for each summarized map $\boldsymbol{m}_j^a$ in the cache $\mathcal{M}^r$, which is described in Algorithm 18 and used in Algorithm 17 for subtracting contributions from each summarized map.

The only complication in this process is ensuring that when summarizing over a set of shared variables Θ_s^r is the antifactors constructed from a summarized map $\boldsymbol{m}_j^a$ operate on only variables present in Θ_s^r. I address this problem by performing a local summarization operation on the factors $\tilde{\Phi}_j^a$ from the summarized map to yield a linear density on only overlapping variables. This local summarization is necessary for two reasons: 1) it forces the summarized map size to grow only with the number of local shared variables, rather than all shared variables, and 2) summarization over the augmented local system removes the contributions to $p\left(\Theta_s^r\right)$ on variables not shared, so there is no double-counted information to remove.

The antifactor summarization step imposes an additional requirement on the map summarization: *summarized maps must be fully constrained linear systems with a positive definite information matrix.* This constraint arises from the linear summarization step, which necessarily performs elimination to compute a summarized density, and both QR and Cholesky factorization algorithms require a full rank, positive definite matrix to yield a result. This constraint is trivially held while using the dense, exact summarization techniques in this chapter, but becomes more significant with the introduction of approximate summarization in Chapter 7.

5.3 Communicating Summarized Maps

The communication phase DDF-SAM 2.0 also uses incremental updates to minimize the computational cost of adding summarized maps using the same communication model as in Section 3.3 to maintain scalability while improving online operation performance. DDF-SAM 2.0 replaces the update procedure for adding summarized maps to account for both incremental updates and the replacement of previously fused maps with antifactors.

Algorithm 19 details the addition of a single summarized map $\boldsymbol{m}_j^b$ from a neighboring robot b, while accounting for any previous map information that might have

Algorithm 19 Adding summarized maps to an augmented local map.

```
function addSummarizedMap(m_j^b, M^r, B^{N'_r}, Θ^{N'_r}) :
    initialize Φ̃_update to summarized map factors Φ̃_j^b
    if an older version m_k^b ∈ M^r exists
        for each linear factor φ̃_k^b ∈ factors(m_k^b)
            add antifactor (φ̃_k^b)^{-1} to update Φ̃_update
        remove m_k^b from M^r
    add m_j^b to M^r
    for each new variable θ_i ∈ variables(m_j^b)

        initialize each variable θ_i in Θ^{N'_r} in same frame
    B^{N'_r} ← linearUpdateIncrementalSolver(Φ̃_update, B^{N'_r})
    return (M^r, B^{N'_r}, Θ^{N'_r})
```

been previously added by updating the cache $\mathcal{M}^r$ and downdating the estimate. The approach constructs a linear graph $\tilde{\Phi}_{update}$ that combines the new summarized map information as well as antifactors to negate double-counted information, and performs the same incremental update as in the local processing step (Section 5.1).

The presence of a previous summarized map $\boldsymbol{m}_k^b$ from the neighboring robot b is another scenario in which information double-counting can occur because map summarizations accumulate all locally available information. In this case, the new summarized map $\boldsymbol{m}_j^b$ contains the full contribution from $\boldsymbol{m}_k^b$ and any new information received between timesteps k and j. Whereas DDF-SAM 1.0 would replace summarized maps in a neighborhood graph $\Phi^{N'_r}$and simply re-eliminate the entire system, this approach allows for incremental updates that replace the previous information.

The initialization of variables in this procedure is significant because all new shared variables from the incoming summarized map, that is, new pure neighborhood variables, I add only one variable to the augmented local system, and it is in the local reference frame of the robot r. In comparison to DDF-SAM 1.0, which adds duplicates

for each observing neighbor to account for separate reference frames, the number of variables does not grow as fast in DDF-SAM 2.0, which yields faster performance for updating neighborhood maps. In addition, the entire system is in the local reference frame of the local robot, thereby making the extended sensor horizon more useful and ensuring a locally consistent map.

5.4 Evaluation

In order to evaluate DDF-SAM 2.0 algorithm without the additional complication of managing separate linearization points for each robot, I constructed a simulated experimental scenario where each robot uses the same linearization points in the same reference frame, which effectively decouples the antifactor downdating procedure from the management of nonlinearities. Within this simplified simulation environment, the I can convincingly demonstrate as a proof of concept that DDF-SAM 2.0 is:

- Online: incremental local and neighborhood map updates remain cheap.
- Consistent: produces an internally-consistent augmented map of the environment with trajectory and extended sensor horizon.

However, this experiment is not able to address scalability concerns due the small number of robots present, which will be addressed in the full evaluation using the extended nonlinear version of DDF-SAM 2.0 in Chapter 6.

I evaluated DDF-SAM 2.0 approach in a simulated scenario designed to represent a small team of quadrotor platforms flying outdoors over a field of landmarks, with the ground truth illustrated in Figure 27. In this system, I model three platforms, each equipped with a downwards facing camera, an inertial measurement unit and a GPS system. The camera runs an image-based feature detector, which produces labeled landmark detections. There are two key simplifying assumptions in this scenario:

- All robots share linearization points for all variables, such that whenever a new

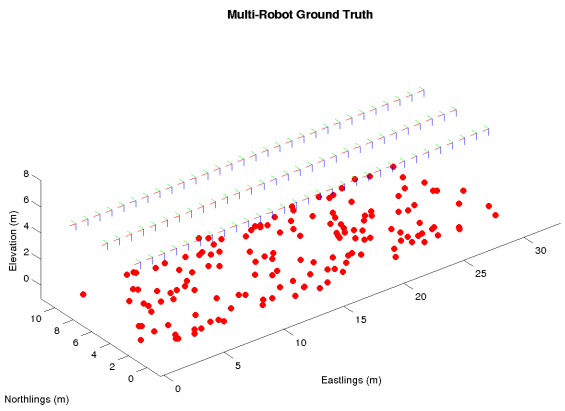


Figure 27: Ground truth for the straight-line visual SLAM scenario, showing three robots flying over a field of randomly generated landmarks, each equipped with downwards-facing a camera.

landmark is observed, the linearization point is shared with all of the other robots, resulting in a nearly-linear system.

- All robots have GPS measurements fixing their position in a global reference frame. By constraining the position of each pose in an absolute coordinate frame, I ensure that rotational drift on each platform cannot result in large changes to the solution in the event of a loop closure

5.5 Results

I ran experiments that simulated experiments that show convincingly that DDF-SAM 2.0 is capable of online operation with internally consistent augmented local map solutions that extend the sensor horizon of individual robots.

The solution for the augmented local maps for each of the robots in the simulated environment is shown in Figure 28, validating the claim that DDF-SAM 2.0 can produce internally consistent maps combining both local and neighborhood information. The solutions show that the maps recover the trajectories of each robot, in addition to the extended sensor horizon in the form of landmarks only observed through neighboring robots.

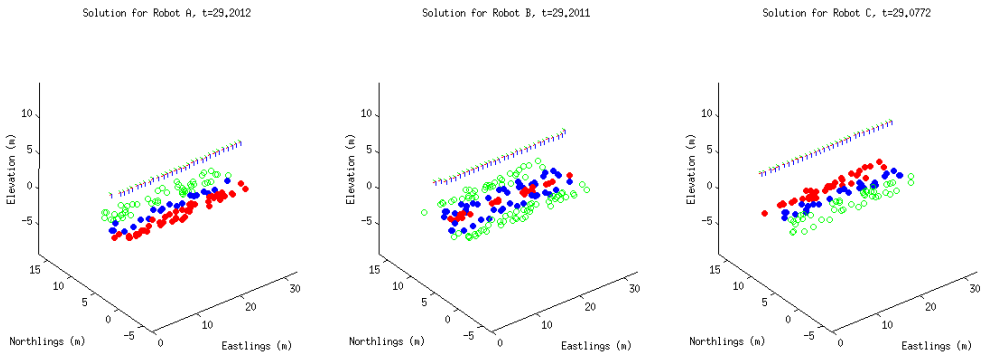


Figure 28: Rendering of the DDF-SAM solution for each robot in straight line simulation, with variables color-coded by membership as red, blue and green, where the groups are pure local, overlapping, and pure neighborhood landmarks, respectively.

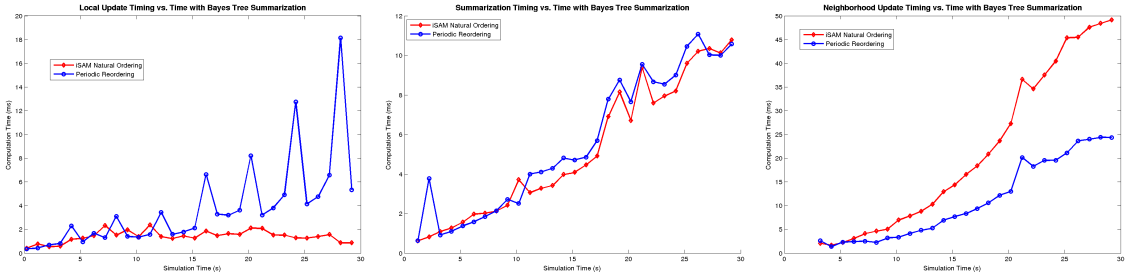


Figure 29: Timing results for operations in simulated straight-line example for local updates (left), summarization (center) and neighborhood updates, showing the compute for Bayes tree exact summarization under different iSAM reordering modes.

To evaluate computational performance in an incremental environment based around a single Bayes tree solver, I plotted timings for the three core DDF-SAM operations in Figure 29. These results use the simplified iSAM algorithm in which all measurements are only linearized once, with a variation in the use periodic variable reordering. In this case, local updates grow at a rate comparable in performance to a single-robot system, and map summarization continues to grow over time, but the slow growth rate in the computational expense of adding summarized maps to the system is an improvement over previous batch optimization techniques.

5.6 Discussion

In this chapter, I introduced DDF-SAM 2.0 as incremental DDF-SAM approach that employs antifactor downdating of map summarizations to avoid double-counted information while blending both local and neighborhood information into augmented local maps. In this chapter, I showed that it is possible to exactly subtract information contributions in order to both remove double-counted information as well as replace previous estimates. The resulting system, using incremental map updates, allows for online operation without the large costs of batch nonlinear optimization.

While I achieved the core goals of combining local and neighborhood information in a single, consistent SLAM solution using an incremental solver, this approach needs to be extended to handle cases where robots do not have GPS-style global measurements and shared linearization points to be generally applicable to realistic robot platforms. Furthermore, the system needs a full evaluation using a larger scale environment to more convincingly demonstrate performance in comparison to pure local mapping, centralized multi-robot mapping and DDF-SAM 1.0.

Chapter 6 extends DDF-SAM 2.0 for a full nonlinear environment where robots

operate in unknown local reference frames, and Chapter 7 provides additional extensions to the summarization approach, which make DDF-SAM 2.0 a viable decentralized mapping approach.

Chapter 6

NONLINEAR DDF-SAM 2.0: INCREMENTAL SAM WITH RELINEARIZATION

In this chapter, I present nonlinear DDF-SAM 2.0, which extends approach in Chapter 5 by directly managing the presence of separate and unknown linearization points for each robot, and performs a full evaluation through large scale simulation. I introduce an approach for performing *relinearization* of previously linearized factors, which solves the problem of combining factors in separate reference frames. Nonlinear DDF-SAM 2.0 has the following desirable characteristics:

- **Online**: Both local and neighborhood updates are incremental, yielding fast update times, and there is no duplication of variables as in the DDF-SAM 1.0 approach, which minimizes the problem size.
- **Scalable**: This approach maintains the same scalability characteristics of previous DDF-SAM techniques through summarized map caching and fusion neighborhood bounding.
- **Consistent**: By actively relinearizing summarized maps, it is possible to achieve the level of estimation error as a DDF-SAM 1.0 approach.
- **Flexible**: Most importantly, nonlinear DDF-SAM 2.0 relaxes constraints on local reference frames and variable initialization, which broadens its applicability to the full range of evaluation scenarios.

The following sections introduce components of the nonlinear DDF-SAM architecture, with an overview provided in Algorithm 20 and the details of the core DDF-SAM

Algorithm 20 Nonlinear DDF-SAM 2.0 formulation incorporating relinearization.

initialize: for each robot r

$\Theta^{N'_r} \leftarrow \{x^r_{init}\}$ ▷ Initialize robot at local origin
$\Phi^{N'_r} \leftarrow \emptyset$ ▷ Nonlinear factor graph
$\mathcal{B}^{N'_r} \leftarrow \emptyset$ ▷ Bayes tree (neighborhood)
$\mathcal{M}^r \leftarrow \emptyset$ ▷ Empty map cache

for each robot r at each discrete timestep k

$\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right) \leftarrow localMeasurements\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}, k\right)$
$\Theta^r_s \leftarrow chooseSharedVariables\left(\Theta^r\right)$
if k is a summarization interval and $|\Theta^r_s| > d_{min}$
$\boldsymbol{m}^r_k \leftarrow summarize\left(\mathcal{B}^{N'_r}, \Theta^r_s, k, \mathcal{M}^r\right)$
add $\boldsymbol{m}^r_k$ to cache $\mathcal{M}^r$ and any replace older $\boldsymbol{m}^r_{k-1}$
$\left(\mathcal{M}^r, \mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right) \leftarrow comms\left(\mathcal{M}^r, \mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right)$

operations of local measurement processing, summarization and communication in Sections 6.1, 6.2 and 6.3, respectively. The general algorithm remains the same as the previously described DDF-SAM implementations, with the main distinction from DDF-SAM 2.0 being a nonlinear factor graph $\Phi^{N'_r}$ that is maintained to allow for relinearization.

The key contribution of this work is the two-stage incremental optimization approach used to solve for both the reference frame transformations and the augmented local map, which rather than using a batch solving technique, iterates across timesteps for online use. This incremental two-stage approach offers the near-constant time updates of an incremental solver. Within the context of DDF-SAM, I use this two-stage optimization technique for fusing both local measurements and factors from incoming summarized maps from neighboring robots.

Because each robot r in the system maintains its own local reference frame, fusing incoming summarized maps requires estimating a reference frame transform T^a_r to bring the summarized map $\boldsymbol{m}^a_j$ from neighboring robot a into the local frame. With an estimate for the reference transformation, I *relinearize* the summarized map to bring it into the local reference frame of the robot in order to do joint inference

over both local and neighborhood map information. Note that this relinearization of incoming summarized map factors is not the same as relinearization of standard nonlinear factors: this relinearization procedure starts with an already linearized factor $\tilde{\phi}^a(\Delta^a)$ and expresses it in the local frame r as $\tilde{\phi}^{a,r}(\Delta^r)$.

The two stages of the optimization process are:

1. **Transform Optimization:** Optimize reference frame transformations as rigid body transformations between constellations of shared variables.

2. **Map Optimization:** Optimize for the map with both local and neighborhood map information with reference frames held constant by transforming any factors from other robots into the frame of the local robot.

While a typical multi-stage approach for the problem would iterate between these stages until convergence at every update, I choose an incremental paradigm such that iterations of each stage roll out over time, which is more suitable for online operation.

6.1 Processing Local Measurements

Within nonlinear DDF-SAM 2.0, I use iSAM 2.0 [84, 85] as the basis for both executing map optimization and determining when to perform a new transform optimization iteration. Because iSAM 2.0 performs relinearization and reordering only when necessary, such as when the linear solution Δ reaches a large enough magnitude, I use this condition to also trigger an iteration of the transform iteration stage. In this section, I introduce this incremental update procedure for the simpler case of updating with locally observed measurements, and expand upon the integration with optimizing over reference frame transformations in Section 6.3. Algorithm 21 presents the overview local measurement processing, derived from the general DDF-SAM formulation from Chapter 3.

Because iSAM 2.0 is a more complicated algorithm, I direct readers to prior work

Algorithm 21 Local processing stage for nonlinear DDF-SAM 2.0 using a nonlinear incremental update to a Bayes tree $\mathcal{B}^{N'_r}$.

`function` $processLocalMeasurements\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}, k\right)$:

`collect measurements` Z_k^r `from local sensors`
`create local factors` Φ_k^r `from` Z_k^r
`for each new factor` $\phi_j^r \in \Phi_k^r$
`for each variable` $\theta_i \in variables\left(\phi_j^r\right)$
`if` θ_i has not been observed
$z_j^r \leftarrow measurement\left(\theta_i^r\right)$
$\theta_i \leftarrow initializeVariable\left(\Theta^{N'_r}, z_j\right)$
`add` θ_i `to` $\Theta^{N'_r}$
$\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right) \leftarrow$
$updateIncrementalSolver\left(\Phi_k, \mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right)$
`return` $\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right)$

---

[85] for the complete details of the approach, and in this section provide a conceptual overview of the algorithm. Algorithm 22 presents the core incremental update procedure for iSAM 2.0, which largely derives from the linear update procedure in Algorithm 15 with regard to isolating the section of the Bayes tree affected by new variables and then constructing and solving a smaller elimination problem before reconstructing the Bayes tree. The primary changes in the approach are as follows:

- Store nonlinear graph: Adds a full nonlinear factor graph $\Phi^{N'_r}$ to the system state $\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right)$, which is necessary to allow for factors to be relinearized as the solution changes.

- Fluid relinearization: As detailed in Algorithm 23, this performs partial relinearization on only those variables that have changed significantly, which prevents the linear system moving too far away from its linearization point by updating the linearization point.

- Partial reordering: When constructing the affected portion of the system as

Algorithm 22 Updating an iSAM 2.0 solver with partial reordering and relinearization at time k with a new set of factors Φ_k, assuming new variables have been already initialized in $\Theta^{N'_r}$. Any antifactors necessary can be optionally added as $\tilde{\Phi}_{antifactors}$, which are necessary during neighborhood updates to replace old summarized maps.

`function` $incrementalUpdate\left(\Phi_k, \tilde{\Phi}_{antifactors}, \mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right)$

`add new factors` Φ_k `to` $\Phi^{N'_r}$ `[NEW]`
$\mathcal{B}_{orphans} \leftarrow \emptyset$
$\Theta_k \leftarrow variables\left(\Phi_k\right)$ ▷ `affected variables`
$\tilde{\Phi}_{update} \leftarrow \tilde{\Phi}_{antifactors}$
$\tilde{\Phi}_{update} \leftarrow liquefyBayesTree\left(\mathcal{B}^{N'_r}, \Theta_k\right)$
$\left(\Theta^{N'_r}, \tilde{\Phi}_{relin}\right) \leftarrow fluidRelinearize\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right)$ `[NEW]`
$\tilde{\Phi}_k \leftarrow linearize\left(\Phi_k, \Theta^{N'_r}\right)$
`for each affected variable` $\theta_j \in \Theta_k \bigcup \Theta_{relin}$
    `remove clique` $\mathcal{C}\left(\theta_j\right)$ `and path to root in` $\mathcal{B}^{N'_r}$
`add orphaned subtrees of` $\mathcal{B}^{N'_r}$ `to` $\mathcal{B}_{orphans}$
`add linear factors` $\tilde{\Phi}_k$ and $\tilde{\Phi}_{relin}$ to $\tilde{\Phi}_{update}$
`compute fill-reducing variable ordering` $\mathcal{P}$ `over` $\tilde{\Phi}_{update}$ `[NEW]`
$\tilde{\Phi}_{update} \leftarrow reorderVariables\left(\tilde{\Phi}_{update}, \mathcal{P}\right)$
$\mathcal{B}^{N'_r} \leftarrow multifrontalEliminate\left(\tilde{\Phi}_{update}\right)$
`insert orphans` $\mathcal{B}_{orphans}$ `into` $\mathcal{B}^{N'_r}$
`return` $\left(\mathcal{B}^{N'_r}, \Theta^{N'_r}, \Phi^{N'_r}\right)$

Algorithm 23 Partial relinearization within iSAM 2.0, which finds a set of nonlinear factors Φ_{relin} to relinearize based on the size of the update Δ for variables Θ_{relin}.

```
function fluidRelinearize(B^{N'_r}, Θ^{N'_r}, Φ^{N'_r})
    compute Δ by backsubstituting B^{N'_r}
    Θ_relin ← ∅
    for each variable θ_i ∈ Θ^{N'_r}
        get corresponding δ_i from Δ for θ_i
        if ‖δ_i‖ > thresh
            update θ_i ← θ_i ⊕ δ_i
            update version θ_i of  in Θ^{N'_r}
            add θ_i to Θ_relin
    find factors Φ_relin in Φ^{N'_r} affected by Θ_relin
    Φ̃_relin ← linearize(Φ_relin, Θ^{N'_r})
    return (Θ^{N'_r}, Φ̃_relin)
```

a graph $\tilde{\Phi}_{update}$, computes a fill-reducing ordering to maintain sparsity in the Bayes tree, which prevents performance from degrading substantially over time.

These changes over the simpler single-linearization approach from Chapter 5, yield both performance update improvements and improved resilience to nonlinearity, which allows for better online performance with better solution quality. The alternative approach for performing reordering and relinearization is to linearize and eliminate the entire system at intervals, which produces update performance characterized by fast, near-constant time updates with periodic large spikes in timing whenever the system is re-solved in batch. Incorporating relinearization into the process also allows the system to handle larger changes in its linearization point, as might occur during a large loop closure.

6.2 Summarizing Local Map Information

While the underlying incremental solving technique has changed to incorporate partial relinearization and reordering, I use the same summarization technique of summarizing from the Bayes tree with antifactor downdating, which maintains the same online performance characteristics as the the earlier version of DDF-SAM 2.0.

The largest difference in map summarization from DDF-SAM 2.0 is that now each robot has a separate local reference frame, and the linear factors comprising each summarized map exists around a different linearization point. The next section will address the process of fusing these summarized maps while managing separate reference frames, in which I apply a transformation to bring an incoming summarized map $\boldsymbol{m}_j^a$ from the robot a into the reference frame of the local robot r.

As a comparison implementation for Bayes tree map summarizations, I employ the pure local dense summarization technique from DDF-SAM 1.0. This starts from the pure local nonlinear factors $\Phi^r \subset \Phi^{N'_r}$, linearized around the current linearization point $\Theta^{N'_r}$ and computes the joint over the shared variables $p\left(\Theta_s^r\right)$. This comparison approach should yield slower summarization than starting from the Bayes tree due to the need to linearize and eliminate the full local system.

6.3 Communicating Summarized Maps

The communication model in nonlinear DDF-SAM 2.0 retains the same structure as the general DDF-SAM model detailed in Section 3.3, and this section focuses on the core data fusion operation of incorporating an incoming summarized map $\boldsymbol{m}_j^a$ from a neighboring robot a at time j into the incremental solver for the local robot r. This approach manages both consistent data fusion, as in DDF-SAM 2.0 (see Section 5.3), as well as the two-stage incremental optimization necessary for joint optimization of the map and reference frame transforms. The resulting algorithm maintains the online performance of an incremental solving technique, while allowing separate and

Algorithm 24 Addition of a summarized map $\boldsymbol{m}_j^a$ from neighboring robot a in a nonlinear DDF-SAM 2.0 system for a local robot r.

`function` $addSummarizedMap\left(\boldsymbol{m}_j^a, \mathcal{M}^r, \mathcal{B}^{N'_r}, \Phi^{N'_r}, \Theta^{N'_r}\right)$:

$\Phi_{update} \leftarrow \emptyset$
$\tilde{\Phi}_{antifactors} \leftarrow \emptyset$
$\Theta_s^a \leftarrow variables\left(\boldsymbol{m}_j^a\right)$
`if any previous map` $\boldsymbol{m}_p^a \in \mathcal{M}^r$ `where` $q < j$
  `for each` $\tilde{\phi}_j^a \in factors\left(\boldsymbol{m}_p^a\right)$
    `add antifactor` $\left(\tilde{\phi}_j^a\right)^{-1}$ `to` $\tilde{\Phi}_{antifactors}$
    `remove factor` $\hat{\phi}_j^a$ `from graph` $\Phi^{N'_r}$
  `remove` $\boldsymbol{m}_p^a$ `from cache` $\mathcal{M}^r$
`add new summarized map` $\boldsymbol{m}_j^a$ `to cache` $\mathcal{M}^r$
$\Theta^{N'_r} \leftarrow transformVariables\left(\Theta^{N'_r}, \Theta_s^a\right)$ `[NEW]`
`for each` $\tilde{\phi}_k^a \in factors\left(\boldsymbol{m}_j^a\right)$
  $\hat{\phi}_k^{a,r} \leftarrow createRelinearizingFactor\left(\tilde{\phi}_k^a, \Theta_s^a\right)$ `[NEW]`
  `add` $\hat{\phi}_k^a$ `to update` Φ_{update}
$\left(\mathcal{B}^{N'_r}, \Phi^{N'_r}, \Theta^{N'_r}\right) \leftarrow$
$incrementalUpdate\left(\Phi_{update}, \tilde{\Phi}_{antifactors}, \mathcal{B}^{N'_r}, \Phi^{N'_r}, \Theta^{N'_r}\right)$
`return` $\left(\mathcal{M}^r, \mathcal{B}^{N'_r}, \Phi^{N'_r}, \Theta^{N'_r}\right)$

---

unknown reference frames and linearization points.

6.3.1 Algorithm Overview

The overall structure of the process for adding new summarized maps in nonlinear DDF-SAM 2.0, detailed in Algorithm 24, follows the same overall structure as the previously described version in Algorithm 19, with additions to handle initializing reference frame transforms and accounting for separate linearization points. As in DDF-SAM 2.0, adding a summarized map $\boldsymbol{m}_j^a$ to an incremental solver can treated as the construction of a set of update factors Φ_{update} that can be added using the same update algorithm used for local measurement updates. In the case where previously observed summarized map $\boldsymbol{m}_p^a$ from the robot a (for timestep $p < j$) exists in the

cache of robot r, this algorithm removes the contributions for any previously added summarized map from both the full neighborhood factor graph $\Phi^{N'_r}$ and from the neighborhood Bayes tree $\mathcal{B}^{N'_r}$ through the addition of antifactors during an iSAM update.

The two key additions appearing in Algorithm 24 are as follows:

- *Transforming Variables*: Compute the reference frame transformation T_r^a between the local frame r and incoming summarized map a, and store this value $\Theta^{N'_r}$. Given the reference frame transform T_r^a, compute and store in $\Theta^{N'_r}$ local versions of each incoming variable.

- *Transforming Factors*: Create a nonlinear *relinearizing factor* for each incoming linear factor $\tilde{\phi}_k^a$ in $\boldsymbol{m}_j^a$, denoted as $\hat{\phi}^{a,r}$, which encodes a linear factor $\tilde{\phi}^a$ and the original linearization point Θ_s^a that can be expressed in the reference frame of r when supplied with a reference frame transform T_r^a. This relinearization only occurs whenever either the transform T_r^a changes or when triggered during fluid relinearization (Algorithm 23) during an incremental update.

In relation to the two stages of optimization within Nonlinear DDF-SAM 2.0, the variable transformation step encompasses the first stage of transform optimization, and the factor transformation step creates the means by which the map optimization stage of optimization freezes the values of the reference frame transformations. The relinearizing factors are part of the map optimization problem and are only parametrized on the reference frame transformations, so that the transformation variables are never explicitly incorporated as a part of the neighborhood Bayes tree $\mathcal{B}^{N'_r}$.

The rest of this section details these two portions of the algorithm, starting with transforming variables in Section 6.3.2 followed by the creation of relinearizing factors in Section 6.3.3. The majority of Section 6.3.3 focuses on the process of performing *relinearization*, in which I express a previously linearized factor in a separate reference

Algorithm 25 Procedure for computing the reference frame transformation and initializing variables.

```
function transformIncomingVariables (Θ^{N'_r}, Θ^a_s)
    T^a_r ← computeTransform (Θ^{N'_r}, Θ^a_s)
    add transform T^a_r to Θ^{N'_r}
    for each new variable θ^a_i ∈ Θ^a_s
        θ^{a,r}_i ← (T^a_r)^{-1} ⊗ θ^a_i      ▷ Express θ^a_i in frame of r
        add θ^{a,r}_i to Θ^{N'_r}
    return Θ^{N'_r}
```

frame without knowledge of the underlying nonlinear representation. Section 6.3.3.3 describes how to integrate the relinearization process into incremental updates.

6.3.2 Transforming Incoming Variables

This stage of incorporating incoming summarized maps, detailed in Algorithm 25, computes both the frame of reference variable T_r^a for a given incoming summarized map $\boldsymbol{m}_j^a$, and initializes versions of the incoming variables in the current neighborhood variable set $\Theta^{N'_r}$.

I compute reference frame transform T_r^a in $computeTransform\,()$ via Algorithm 12, which employs SVD factorization over matching landmark pairs to compute a least-squares solution for rigid-body alignment. This procedure for computing the reference frame transform could be extended to incorporate additional variable types, such as shared robot poses. In addition, data association algorithms such as RANSAC, which would be necessary if the shared variables did not have globally unique labels, typically compute a solution for the rigid transform as a part of determining correspondences. I provide this simpler version in this section as a reference implementation.

Using the reference frame transform, I initialize versions of each variable θ_i^a in Θ_s^a, which I denote as $\theta_i^{a,r}$, which expresses the variable θ_i^a in reference frame r. To

generalize the reference frame transformation operation, I denote the transformation $\theta_i^r = T_a^r \otimes \theta_i^a$, which is either a Lie group action of the transform T_a^r on a landmark θ_i^a or a Lie group composition operation on θ_i^a for poses.

It should be noted in this algorithm that I also add the reference frame transform T_r^a to the full set of neighborhood values $\Theta^{N'_r}$, even though the map optimization stage will not compute updates for T_r^a, to allow the reference frame transform to be available during fluid relinearization during updates.

6.3.3 Transforming Incoming Factors

I introduce in this section a technique that expresses an incoming linear factor $\tilde{\phi}^a(\Delta^a)$ from neighboring robot a (which has a separate reference frame) and expresses the linear factor $\tilde{\phi}^{a,r}(\Delta^r)$ within the reference frame of the local robot r to allow for optimization within a consistent reference frame. This approach ensures correct data fusion between separate frames, while maintaining the performance improvements inherent in an incremental update procedure.

The key to this approach is that the majority of the error between constellations of variables in separate reference frames can be accounted for by a rigid transformation between the reference frames. This property enables the use of a relinearization factor $\hat{\phi}^{a,r}$, which encapsulates the original linear factor $\tilde{\phi}^a$ and its original linearization point Θ^a, to act as a nonlinear factor within the context of updates for iSAM 2.0. The actual relinearization procedure occurs anytime a relinearizing factor $\hat{\phi}^{a,r}$, added to a robot r from a summarized map $\boldsymbol{m}_j^a$, is linearized during an incremental update step $incrementalUpdate()$.

I will derive this relinearization procedure first under the assumption that the reference frame transformation T_r^a exactly accounts for all difference between the variables of Θ^a and Θ^r in Section 6.3.3.1, and then relax this exactness assumption in Section 6.3.3.2.

6.3.3.1 Transform Composition for Relinearization

I present in this section the core requirements for relinearization of a previously linearized factor $\tilde{\phi}^a(\Delta^a)$ in a new reference frame r given the linearization points in each frame (Θ^a and Θ^r, respectively) and a transform between these reference frames T_r^a. I will derive this formulation starting from a hypothetical nonlinear factor $\phi^a(\Theta^a)$ and show how I can compute an new relinearized factor $\tilde{\phi}^{a,r}(\Delta^r)$ in the reference frame r without knowledge of the ϕ^a.

For an arbitrary factor $\phi^a(\Theta^a)$ fixed in the reference frame of robot a which linearizes to the linear factor $\tilde{\phi}^a(\Delta^a)$, I can map evaluate its error at Θ^r using a known reference frame transformation, where

$$\Theta^a = T_r^a \otimes \Theta^r, \tag{20}$$

yielding a factor formulation via composition $\phi^a(T_r^a \otimes \Theta^r)$ in the original reference frame a. In this instance, I denote applying a transformation T_r^a to a set of variables Θ^r to be the equivalent of applying the transform to each variable $\theta_i^r \in \Theta^r$ that yields a new set of transformed variables Θ^a. In this case, there is always a one-one mapping between a variable $\theta_i^r \in \Theta^r$ and $\theta_i^a \in \Theta^a$, such that $\theta_i^a = T_r^a \otimes \theta_i^r$.

A nonlinear factor ϕ^a operating on the variables Θ^a from the incoming robot a is

$$\phi^a(\Theta^a) \triangleq \|h(\Theta^a) - z\|_\Sigma^2, \tag{21}$$

with a nonlinear measurement prediction function $h : \Theta^a \to Z$, where z is the measured value for the given factor, Σ is the measurement covariance.

The goal in this step is to have a nonlinear factor as a function of Θ^r which gives exactly the same error as original factor ϕ^a after we transform Θ^r into the original a frame. I denote this new transformed factor as $\phi^{a,r}(\Theta^r)$, which is the factor ϕ^a in the frame of a expressed in the frame of r. In this formulation, I assume that the the transformation T_r^a is exact, i.e., $\Theta^a = T_r^a \otimes \Theta^r$, and express the transformed factor as

$$\phi^{a,r}(\Theta^r) \triangleq \|h^{a,r}(\Theta^r) - z\|_\Sigma^2.$$

I can use the reference frame transformation to express the transformed measurement function $h^{a,r}(\Theta^r)$ in terms of the original measurement function $h(\Theta^a)$, as follows

$$h^{a,r}(\Theta^r) \triangleq h(T_r^a \otimes \Theta^r) = h(\Theta^a), \tag{22}$$

which transforms Θ^r into the a frame before evaluating h.

To optimize, I actually need a linearized version of the transformed factor $\phi^{a,r}$ evaluated at Θ^r, which I compute as

$$\tilde{\phi}^{a,r}(\Delta^r) \triangleq \left\| \frac{\partial h^{a,r}(\Theta^r)}{\partial \Theta^r} \Delta^r + h^{a,r}(\Theta^r) - z \right\|_\Sigma^2,$$

and expand $\frac{\partial h^{a,r}(\Theta^r)}{\partial \Theta^r}$ via the chain rule to form

$$\tilde{\phi}^{a,r}(\Delta^r) \triangleq \left\| h'(T_r^a \otimes \Theta^r) \frac{\partial T_r^a \otimes \Theta^r}{\partial \Theta^r} \Delta^r + h(T_r^a \otimes \Theta^r) - z \right\|_\Sigma^2. \tag{23}$$

Substituting in the transformed terms from eq. 22 yields

$$\tilde{\phi}^{a,r}(\Delta^r) \triangleq \left\| h'(\Theta^a) \frac{\partial T_r^a \otimes \Theta^r}{\partial \Theta^r} \Delta^r + h(\Theta^a) - z \right\|_\Sigma^2, \tag{24}$$

in which only difference between $\tilde{\phi}^{a,r}$ and the original linearization $\tilde{\phi}^a$ is the Jacobian matrix expressing $\frac{\partial T_r^a \otimes \Theta^r}{\partial \Theta^r}$ that is square in the dimension d of Θ^r. For convenience of notation, let the matrix $J^{a,r} \triangleq \frac{\partial T_r^a \otimes \Theta^r}{\partial \Theta^r}$.

I can decompose the transformation operation acting on a set of variables $T_r^a \otimes \Theta^r$ into a set of independent transformations acting on individual variables $\theta_i^r \in \Theta^r$, such that each operation is $T_r^a \otimes \theta_i^r$, which yields a similar decomposition of $J^{a,r}$ as a block-diagonal matrix,

$$J^{a,r} \triangleq \begin{bmatrix} J_1^{a,r} & & & & \\ & J_2^{a,r} & & & \\ & & \ddots & & \\ & & & J_{M-1}^{a,r} & \\ & & & & J_M^{a,r} \end{bmatrix}$$

for M variables, where each block $J_i^{a,r} \triangleq \frac{\partial T_r^a \otimes \theta_i^r}{\partial \theta_i^r}$. Each block $J_i^{a,r}$ is square in the dimension d_i of the variable θ_i^r.

Computing this Jacobian block depends on the type of variable θ_i^r due to the relationship between the

- *Landmarks:* In the case of a landmark variable θ_i^r ($\theta_i^r \in \mathbb{R}^2$ or $\theta_i^r \in \mathbb{R}^3$), the transformation operation $T_r^a \otimes \theta_i^r$ is an Lie group action of the transform T_r^a on the point θ_i^r. The partial derivative of this group action on a point in terms of the point θ_i^r, is the Jacobian block $J_i^{a,r} \triangleq R_r^a$, where R_r^a is the rotation component of the transformation T_r^a.

- *Poses*: Where θ_i^r is a pose ($\theta_i^r \in \mathcal{SE}(2)$ or $\theta_i^r \in \mathcal{SE}(3)$), the transformation $T_r^a \otimes \theta_i^r$ is a Lie group composition operation, for which the partial derivative in terms of the θ_i^r is the Jacobian block $J_i^{a,r} \triangleq \mathcal{I}$.

As a simple example, consider a factor operating on four variables, where each variable is a 3D point, yields a 12x12 block diagonal matrix for $J^{a,r}$

$$J^{a,r} = \begin{bmatrix} R_r^a & & & \\ & R_r^a & & \\ & & R_r^a & \\ & & & R_r^a \end{bmatrix},$$

where each block $J_i^{a,r}$ is the rotation matrix component R_r^a from the transform T_r^a. In this case, the update vector is $\Delta^r \in \mathbb{R}^{12}$, and has a structure such that there is a sub-vector $\delta_i^r \in \mathbb{R}^3$ corresponding to each variable θ_i^r, laid out as

$$\Delta^r \triangleq \begin{bmatrix} \delta_1^r \\ \delta_2^r \\ \delta_3^r \\ \delta_4^r \end{bmatrix}.$$

The transform Jacobian $J^{a,r}$ projects the update vector Δ^r in the frame r to a corresponding update vector Δ^a in the frame of a.

6.3.3.2 Residual Correction

In practical cases, however, the reference frame transformation $\Theta^a = T_r^a \otimes \Theta^r$ is not exact, which implies that

$$\phi^a\left(\Theta^a\right) \neq \phi^a\left(T_r^a \otimes \Theta^r\right)$$

where the error of the original factor $\phi^a\left(\Theta^a\right)$ evaluated at Θ^a is no longer exactly the error of $\phi^a\left(T_r^a \otimes \Theta^r\right)$. In this section, I relax the reference frame transformation assumption by applying a correction to account the change in the error.

While it is possible to simply assume an exact transformation (as in eq. 20), this will result in additional approximation error within the residual term of the linear factor, and by applying a more careful correction, it is possible to improve this approximation. I relax the reference frame transformation relationship from a strict equality, as in eq. 20, to an approximate equality

$$\Theta^a \approx T_r^a \otimes \Theta^r. \tag{25}$$

This approximation can typically be considered to be close to the true value because any computed transform T_r^a should optimize the following cost function

$$e\left(T_r^a\right) \triangleq \frac{1}{2}\left\|\Delta^{r,a}\right\|^2, \tag{26}$$

in which the local error term $\Delta^{r,a}$, where

$$\Delta^{r,a} \triangleq \Theta^a - \left(T_r^a \otimes \Theta^r\right), \tag{27}$$

which expresses the difference between Θ^a and Θ^r in the reference frame of a. Because the variables have a one-one mapping between Θ^a and Θ^r, $\Delta^{r,a}$ has the same structure as the same structure Δ^a or Δ^r. The $computeTransform\left(\right)$ function (Algorithm 12) employed in Section 6.3.2 computes the least-squares solution for this problem.

In realistic mapping scenarios, this error $e\left(T_r^a\right)$ will typically be small, though in practice it is not small enough to assume $e\left(T_r^a\right) = 0$ during relinearization. This error comes from the fact that each robot a and r will have different map solutions constructed from different measurements, and will decrease as Θ^a and Θ^r converge over time, but I will need to account for it to allow for online operation.

Because of the error in the reference frame transformation, the relationship between the measurement functions

$$h\left(T_r^a \otimes \Theta^r\right) \neq h\left(\Theta^a\right)$$

is now not known exactly for use in substitution. As h is a nonlinear function, there is no exact means of evaluating $h\left(T_r^a \otimes \Theta^r\right)$ without having access to the nonlinear function itself on the robot r.

From the relinearized version of the factor from eq. 23, If I assume that $h'\left(T_r^a \otimes \Theta^r\right) \approx h'\left(\Theta^a\right)$, which I can substitute in to yield

$$\tilde{\phi}^{a,r}\left(\Delta^r\right) \triangleq \left\| h'\left(\Theta^a\right) \frac{\partial T_r^a \otimes \Theta^r}{\partial \Theta^r} \Delta^r + h\left(T_r^a \otimes \Theta^r\right) - z \right\|_\Sigma^2$$

in which the remaining term $h\left(T_r^a \otimes \Theta^r\right)$ needs to be resolved.

I can use an linear approximation centered around the linearization point Θ^a to compute $h\left(T_r^a \otimes \Theta^r\right)$ as

$$h\left(T_r^a \otimes \Theta^r\right) = h\left(\Theta^a \oplus \Delta^{r,a}\right) \approx h\left(\Theta^a\right) + h'\left(\Theta^a\right) \Delta^{r,a}, \tag{28}$$

in which $\Theta^a \oplus \Delta^{r,a}$ applies the $\Delta^{r,a}$ as an update to Θ^a using a manifold retraction, which is possible because $\Delta^{r,a}$ is expressed in the tangent space of Θ^a. Substituting this approximation into the original formulation yields a fully relinearized factor expressed in Δ^r, that accounts for both the reference frame transformation T_r^a and corrects for error in the transform itself

$$\tilde{\phi}^{a,r}\left(\Delta^r\right) \triangleq \left\| h'\left(\Theta^a\right) \frac{\partial T_r^a \otimes \Theta^r}{\partial \Theta^r} \Delta^r + h\left(\Theta^a\right) + h'\left(\Theta^a\right) \Delta^{r,a} - z \right\|_\Sigma^2. \tag{29}$$

Given these terms, I can use eq. 29 to construct a relinearized factor accounting for the reference frame transformation T_r^a, which can convert an incoming linear factor

$$\tilde{\phi}^a\left(\Delta^a\right) \triangleq \left\|A\Delta^a - b\right\|_\Sigma^2 \tag{30}$$

received from a neighboring robot a, expressed as a function of Δ^a, where A is the measurement Jacobian $h'\left(\Theta^a\right)$ and $b \triangleq h\left(\Theta^a\right) - z$, into a factor that is a function of r

$$\tilde{\phi}^{a,r}\left(\Delta^r\right) \triangleq \left\|AJ^{a,r}\Delta^r - b + A\Delta^{r,a}\right\|_\Sigma^2, \tag{31}$$

which can be further condensed into the same form as eq. 30 as

$$\tilde{\phi}^{a,r}\left(\Delta^a\right) \triangleq \left\|A'\Delta^a - b'\right\|_\Sigma^2,$$

where $A' \triangleq AJ^{a,r}$ and $b' = b - A\Delta^{r,a}$.

6.3.3.3 Incorporating Relinearization

It is important to note that the relinearization operation in the previous section does not occur explicitly as a part of when a robot fuses an incoming summarized map, but rather whenever the factors comprising the summarized map are relinearized. To facilitate this delayed relinearization, I create relinearizing factors $\hat{\phi}^{a,r}\left(\Theta^{N'_r}\right)$, which act variables in the frame of r and encapsulate both the original linear factor $\tilde{\phi}^a$ from robot a and its linearization point Θ^a, and add $\hat{\phi}^{a,r}$ to the full nonlinear factor graph $\Phi^{N'_r}$. Whenever relinearizing factor $\hat{\phi}^{a,r}$ is linearized around $\Theta^{N'_r}$, it computes a new $\Delta^{r,a}$ using the current values for Θ^r in $\Theta^{N'_r}$, and returns $\tilde{\phi}^{a,r}$ as defined by eq. 29.

6.4 Evaluation

To evaluate nonlinear DDF-SAM 2.0 (and, by extension, the core contributions of DDF-SAM 2.0 as presented in Chapter 5), I used the experimental procedure developed previously (See section 3.4), which correctly evaluates the following claims for DDF-SAM:

- Online: I compute timings of each core DDF-SAM operation (local update, summarization, neighborhood update), which show that performance remains tractable.

- Scalable: I completed each test scenario under different values for the fusion neighborhood bound K, which demonstrates that it is possible to bound communication and computational growth with the addition of new robots.

- Consistent: I evaluate map correctness using the error metrics developed in Section 3.4.5, as well as comparing the error in computing transformed frame of reference variables to compare performance with the simultaneous optimization of DDF-SAM 1.0.

As test scenarios, I use the Manhattan-world simulated dataset for qualitative analysis, and the large scale simulation for aggregate statistical analysis. As real-world scenario, I ran the system on the Freiburg dataset (see Section 3.4.6.3). The primary experimental system is the nonlinear DDF-SAM 2.0 algorithm using Bayes tree summarization with antifactors, with comparison implementations of nonlinear DDF-SAM 2.0 with pure local dense summarization (without antifactors), DDF-SAM 1.0 using iSAM 2.0 as the solver for the local map, pure local iSAM 2.0 without multi-robot contributions, and a fully centralized system using iSAM 2.0. The expectation is that the centralized solution is a gold-standard for map quality, as it uses all available measurement information without any approximation.

6.5 Results

I completed the experimental evaluations from the previous section and have results that validate the following claims:

- Online: Computational costs for local and neighborhood updates grow relatively slowly, neighborhood map fusion is dramatically faster than with DDF-SAM 1.0.

- Scalable: The experiments show that the communication costs can be bounded through the use of the fusion neighborhood bound K.
- Consistent: The error results convincingly show that nonlinear DDF-SAM 2.0 produces estimates of usable quality, and compute better estimates of reference frame transforms than DDF-SAM 1.0.
- Extended sensor horizon: The Manhattan-world dataset provides a clear visualization of the impact of using an augmented local map, yielding both an extended sensor horizon and the full robot trajectory in the same map.

I will present the qualitative results first to demonstrate the structural improvements in DDF-SAM 2.0 over a pure local or DDF-SAM 1.0 approach, followed by the quantitative results from the large scale scenario for more detailed analysis.

6.5.1 Manhattan-world Simulation

The benefits of using a DDF-SAM 2.0 over either a pure local SLAM system or DDF-SAM 1.0 are readily apparent in the illustrative Manhattan-world scenario, showing both an extended sensor horizon, but improved local estimates due to information contributed from neighboring robots. Figure 30 shows the pure local map solution for a selected robot in comparison to the solution computed from nonlinear DDF-SAM 2.0. The extended sensor horizon for the robot using DDF-SAM 2.0 is apparent in the additional pure neighborhood variables in the resulting map, as is the robot trajectory, which did not exist as a part of the same estimate in DDF-SAM 1.0.

In addition, examining the covariance ellipses for the landmark marginals shows that the additional information from the neighboring robots improved the certainty of the local estimate as well. This can be seen most clearly in when comparing the marginals landmark marginals at the end of the trajectory, where the landmarks have much larger covariance ellipses than their counterparts in the DDF-SAM 2.0 solution.

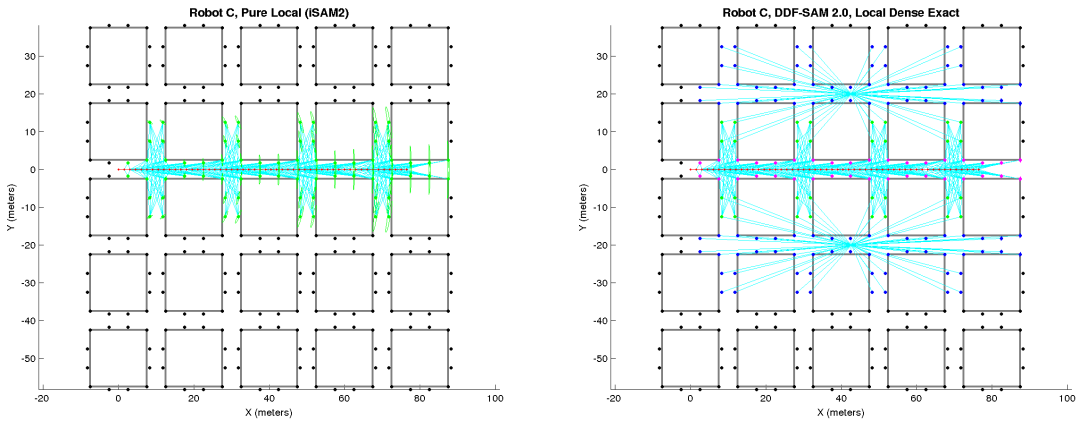


Figure 30: Map solutions with landmark marginals drawn for Manhattan-world simulated dataset for a single robot moving from left to right, showing a purely local solution (left) and the result of nonlinear DDF-SAM 2.0 with two fusion neighbors. Landmarks in the multi-robot example are color-coded in groups for pure local, overlapping, and pure neighborhood variables, as magenta, green and blue, respectively.

Even the variables that were only observed locally in the DDF-SAM 2.0 solution have improved certainty.

6.5.2 Freiburg Dataset

The real-robot scenario in the Freiburg dataset shows how the system can handle data collected from real sensor systems, particularly in the case of long trajectories. As this is only a three-robot dataset, it is not possible to evaluate claims regarding scalability to large team sizes, but it does provide a basis for analyzing the quality of map solutions in a significantly longer dataset.

As an overview of the the sort of map solutions that are feasible within nonlinear DDF-SAM 2.0, Figure 31 shows the results for a single robot within the Freiburg dataset, using both local dense exact summarization as a control, and showing results from the summarization technique starting from the Bayes tree and using antifactors. It should be noted that due to conservative multi-robot data association, there are several variables that have multiple ellipses - these occur when no data association was recorded between these variables originally. In both maps, the solutions are visually identical, and as an improvement to the neighborhood map solution shown in Figure

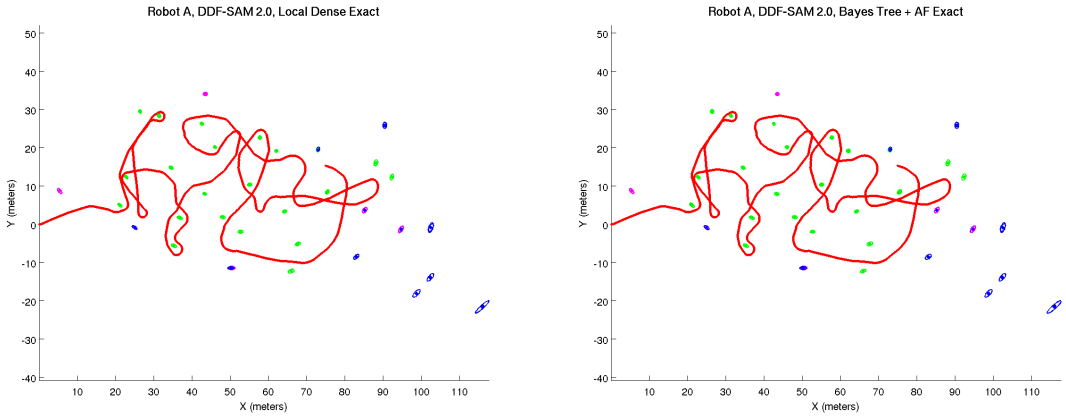


Figure 31: Renderings of solutions from Freiburg dataset run 2, showing the final solution for robot A from both the pure local summarization approach with no antifactors (left) and the solution with antifactors. Both solutions have the marginals for landmarks shown, and use the same color coding as previous renderings.

17, these systems also show the full trajectory of the robots.

Figure 32 provides plots of update performance vs. time, which shows the benefit of using an incremental solver for the full system. In local updates on the Freiburg dataset, nonlinear DDF-SAM 2.0 is only slightly slower than the DDF-SAM 1.0 local mapping solver, while the DDF-SAM 2.0 system is actually solving a larger system that has extra variables from the fused summarized maps. The largest improvement of DDF-SAM 1.0 performance is in fusing new summarized maps, which remains under 15 ms throughout much of the run. This dramatic improvement demonstrates how nonlinear DDF-SAM 2.0 is much more suited for online, realtime operation than its predecessor.

I plot summarization costs, both compute time and transmission size, in Figure 33, which show that summarizing directly from a Bayes tree is substantially faster than computing a summarization by relinearizing and fully eliminating the pure local system. In large part, this is due to the much larger local system present in this scenario as compared to others, in which the cost of eliminating the full system grows rapidly. Note that in this case the DDF-SAM 1.0 and DDF-SAM 2.0 local dense exact approaches are doing equivalent operations, which explains timing that

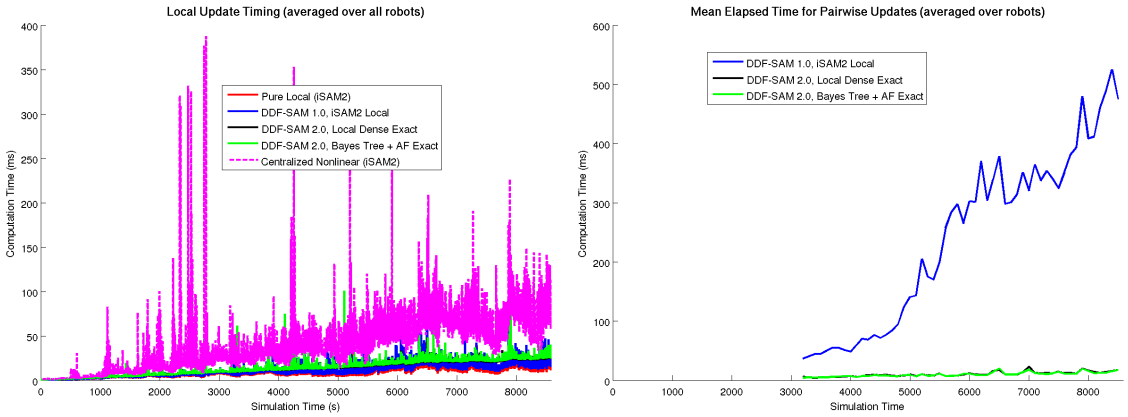


Figure 32: Updating timing from Freiburg dataset run 2, showing the timing for local updates (left) and the mean pairwise update timing for sharing and fusing summarized maps with neighbors. As a comparison, both plots show the timing for DDF-SAM 1.0 using an incremental local solver. In addition, the local timing plot compares against a pure local solver and a centralized solution.

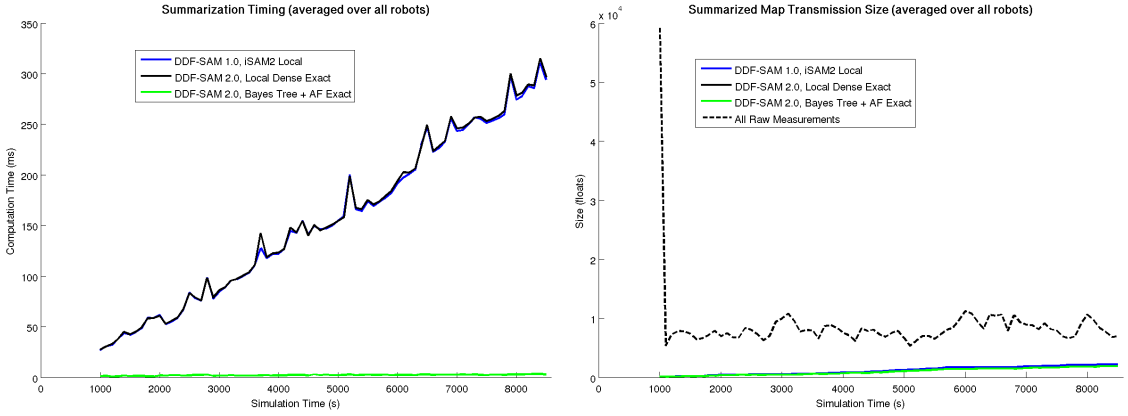


Figure 33: Summarization costs from the Freiburg dataset run 2, showing the time to compute a summarized map (left) and the transmission size of summarized maps.

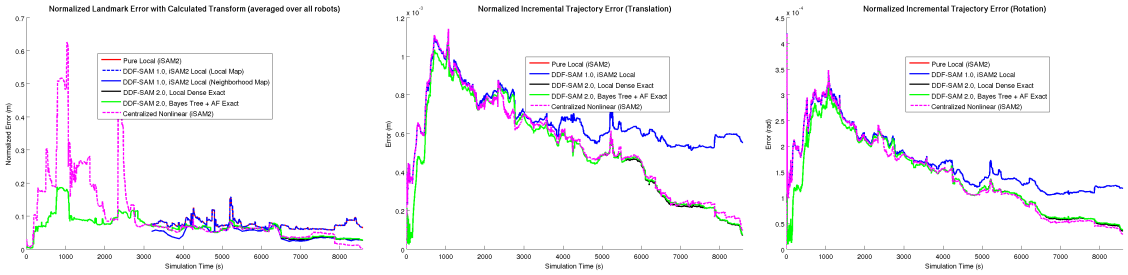


Figure 34: Error in the solutions in comparison to the centralized batch solution for the Freiburg dataset run 2, showing, from left to right, the landmark constellation error, the error in translation for the trajectory, and the error in rotation for the trajectory. Note that error metrics are normalized by the number of variables to allow for comparisons between systems of different sizes. I also compare against pure local, centralized and DDF-SAM 1.0 solutions.

is nearly identical. With regard to map transmission size, all of the techniques for summarization are still exact summarization techniques over the same set of variables, so there is no significant difference in communication cost.

Figure 34 shows plots of the solution error over time as compared to the centralized batch solution, broken out separate error components on the landmarks and the trajectory. Note that the centralized incremental solver in this case has larger error spikes at the beginning of the dataset primarily because the graphs from each robot are disconnected, which allows them to diverge from the true solution. The results for nonlinear DDF-SAM 2.0 each part show convergence to error at the level of the centralized incremental solution, which shows that nonlinear DDF-SAM 2.0 has a consistent solution.

6.5.3 Large Scale Simulation

I ran the the experiments in the large scale simulated environment, designed to evaluate quantitative metrics for performance over a large team of robots, and the results validated the following claims:

- Online: Update performance timing is fast enough for real-time, and neighborhood updates in DDF-SAM 2.0 are dramatically faster than those DDF-SAM

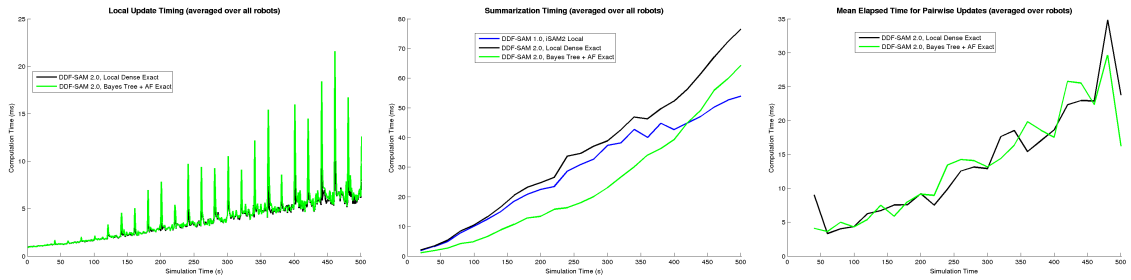


Figure 35: Timing performance for operations, from left to right, local updates, summarization and neighborhood updates for the $K = 4$ scenario. Note that the neighborhood updates measured are for a pairwise sharing operation with another single robot. The experimental versions shown are DDF-SAM 2.0 with Bayes tree summarization and pure local dense summarization, as well as DDF-SAM 1.0 for map summarization.

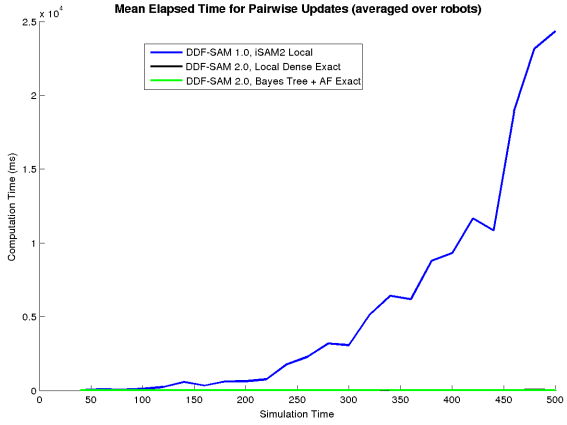


Figure 36: Mean timing for performing a neighborhood pairwise update for the $K = 4$ scenario, showing DDF-SAM 1.0 as a comparison. In the neighborhood update plot, note that the black line for the local dense exact summarization variant is obscured by the green line for nonlinear DDF-SAM 2.0 Bayes tree summarization.

1.0.

- Scalable: The network bandwidth cost has clear striations with K, showing that the fusion neighbor bound is still an effective means of bounding costs.

- Consistent: The map solutions have error that is low enough for reliable use, and better than DDF-SAM 1.0, across all statistics.

Through performance timings, I can convincingly validate claims that nonlinear DDF-SAM 2.0 is fast enough for online operation, and provides a substantial improvement

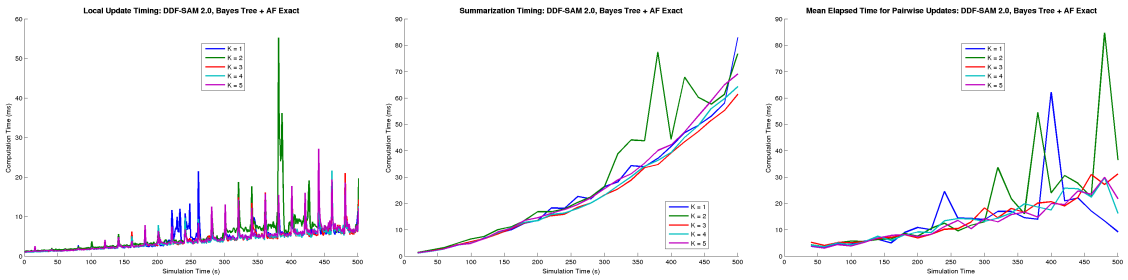


Figure 37: Timing performance for operations for Bayes tree summarization, while varying K values, from left to right, local updates, summarization and neighborhood updates. Note that the neighborhood updates measured are for a pairwise sharing operation with another single robot.

over DDF-SAM 1.0 when comparing the cost of adding summarized maps. Figure 35 shows timing for local updates, summarization and neighborhood updates when comparing the two variations on nonlinear DDF-SAM 2.0. In this case, both the local and neighborhood updates are on par with each other, and grow relatively slowly. Summarization timing validates the expected result that summarizing from the Bayes tree will be faster than from the pure local nonlinear factor graph, but the benefit does not substantially change the growth rate, even compared to DDF-SAM 1.0 summarization. Figure 36 shows the same timing comparison for neighborhood updates, but in comparison to DDF-SAM 1.0. The biggest improvement over DDF-SAM 1.0 occurs is in neighborhood updates, where DDF-SAM 1.0 uses a batch nonlinear optimization technique, and DDF-SAM 2.0 uses an incremental update. I show the timing comparison separately due to the large difference between DDF-SAM 1.0 and DDF-SAM 2.0.

As an additional set of performance metrics, Figure 37 plots the timing for the three main operations split out by the K, the fusion neighborhood bound, which shows that for this scenario there is no strong striations for differing values of K.

However, the neighborhood bound K still has an effect on message sizes between robots, as can be seen in Figure 38, where choosing a larger value for K results in more network traffic. This indicates that it remains possible to scale to large teams of

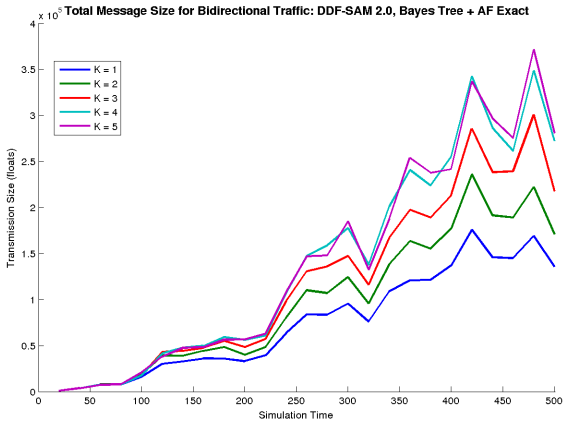


Figure 38: Bidirectional communication bandwidth through a single node, averaged over all robots, varying with the size of the communication neighborhood K for Bayes tree summarization.

robots by limiting the fusion neighborhood of each individual platform. Furthermore, the small difference between the $K = 4$ and $K = 5$ indicates that there is a natural limit to fusion neighborhood size imposed by the overlapping regions in a test scenario with 25 robots.

I evaluate map consistency through map error metrics, plotted in Figure 39, covering local mapping, which convincingly show that the maps produced through nonlinear DDF-SAM 2.0 are sufficiently correct for realistic use. These plots show that both nonlinear DDF-SAM 2.0 approaches yield solutions that, while greater in error than a pure local or centralized solution, are low enough to remain practical with growth trends that do not explode over time. Of particular interest is the comparison of normalized landmark constellation error, which also compares DDF-SAM 1.0, which appears to continue growing in error during the course of the dataset.

As an internal metric for correctness, I examine the estimation error for the frame of reference transforms themselves in Figure 40, which is a useful proxy for determining sources of error in estimation for both DDF-SAM 1.0 and 2.0. It also compares the simultaneous optimization of maps and reference frames of DDF-SAM 1.0 to the

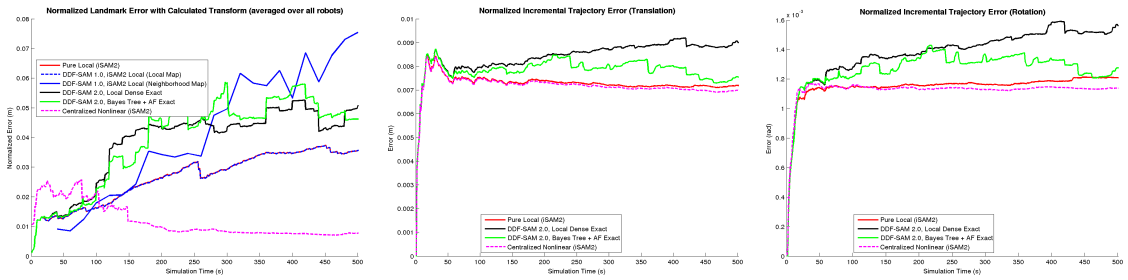


Figure 39: Effect of summarization techniques on solution error metrics (landmark constellation error, trajectory translation, trajectory rotation from left to right) for the large scale simulation with 25 robots and neighborhood bound $K = 4$, showing both exact summarization techniques, as well as naive Bayes and local tree approximations. Comparison techniques in these plots are the single-robot solver, a centralized incremental solution, and DDF-SAM 1.0 for the landmark constellation error. The DDF-SAM 1.0 trajectory error results are not shown as they are identical to the pure local solver.

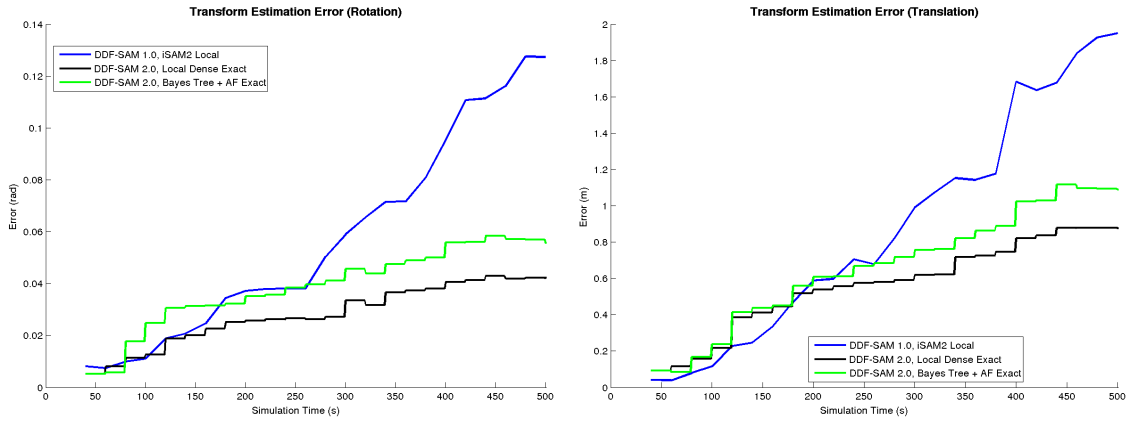


Figure 40: Estimation error for transforms between frames, split into rotation (left) and translation components. Results from DDF-SAM 1.0 on the same transform estimation metrics are shown for comparison.

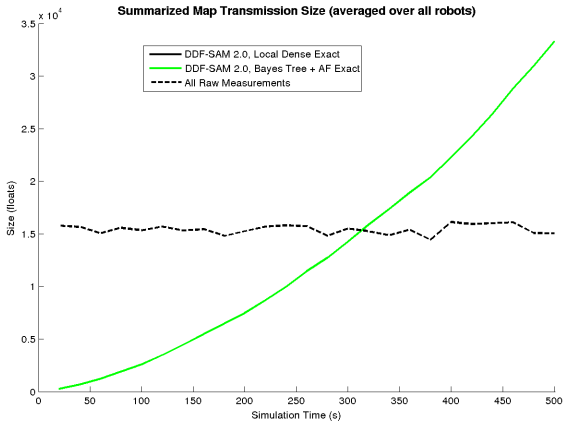


Figure 41: Summarized map transmission size for both nonlinear DDF-SAM 2.0 variations as compared to sending of accumulated nonlinear measurements to a single fusion center. In this scenario, both DDF-SAM 2.0 map summarization techniques have have the same transmission size.

two-phase approach used in DDF-SAM 2.0. As can be seen from the plot, the error increases more quickly for the DDF-SAM 1.0 solution, which is a particularly large problem in the case of rotation, as rotation introduces a large portion of the nonlinearity into the model.

While these results have, thus far, shown significant improvements along the metrics presented, the growth rate in summarized map transmission size continues to be quadratic in the number of variables, which will limit scalability due to communication bandwidth use. Figure 41 plots the growth in transmission size over time, averaged over the full team of robots, which surpasses the transmission cost of sending accumulated nonlinear measurements back to a local server just after half-way through the dataset. While the cost of transmitting the nonlinear measurements does not capture all of the problems inherent in using a centralized map fusion approach, it highlights an area where DDF-SAM needs to improve.

6.6 Discussion

This chapter has introduced a nonlinear extension of DDF-SAM 2.0 that maintains the performance benefits of using an incremental solver from the formulation in Chapter 5, while managing unknown relative reference frames and linearization points. The key contribution enabling this extension is the multi-stage optimization of summarized maps within a state-of-the-art incremental solver (iSAM 2.0), with performance that is as good or better than DDF-SAM 1.0 in the same scenarios. The experimental results demonstrate that nonlinear DDF-SAM 2.0 is online, scalable and consistent, as well being an improvement on DDF-SAM 1.0.

Because both DDF-SAM 1.0 and nonlinear DDF-SAM 2.0 can both solve the same classes of problems - decentralized mapping with unknown relative reference frames - it is useful to compare how the two approaches differ. Both approaches need to solve the problem of establishing relationships between variables linearized in separate reference frames corresponding to the same ground truth landmark. The distinction between the approach used in DDF-SAM 1.0 and the multi-stage incremental optimization approach of nonlinear DDF-SAM 2.0 described in this chapter centers on differing methods of mapping variables between reference frames. DDF-SAM 1.0 uses *hard equality constraints* to relate factors between different reference frames, which is more general, though more complicated approach. For nonlinear DDF-SAM 2.0, I employ *transform composition* to combine transforming variables into a separate reference frame and evaluating a linear error function.

The DDF-SAM 1.0 approach has several problems induced by the additional graph complexity that are rectified in nonlinear DDF-SAM 2.0:

- The constrained factor graph representing the neighborhood graph in DDF-SAM 1.0 includes a duplicate version of each variable appearing in multiple reference frames, for each reference frame, even though they map to a single real-world landmark.

- Because every variable observed by multiple robots in DDF-SAM 1.0 must have a ternary constraint added connected to a frame of reference variable, the frame of reference variables will have a high degree in a graph theoretic sense, which will result in poor performance with direct solving techniques (an Eiffel tower as described in [41]) due to having a large condition number.

The underlying problem with the DDF-SAM 1.0 approach is it takes a far too conservative approach to handling the insertion of linearized factors into a new nonlinear system, in that it assumes knowing the linearized form of a factor in one reference frame tells us nothing about the its linearized form in another frame. Nonlinear DDF-SAM 2.0 solves this problem by directly managing the relinearization of incoming summarized map factors, and combined with antifactor downdating to maintain consistency, is a superior approach in this regard.

While the introduction of relinearization and the use of an incremental solver have yielded large benefits in update performance, there are two key metrics where both DDF-SAM 1.0 and 2.0 have shortcomings that could limit scalability: summarized map transmission size. In all cases presented thus far, summarized maps have been dense joint factors over the full set of shared variables Θ_s^r, and the size of the resulting information matrices transmitted between robots grows quadratically with the number of variables. The next chapter addresses this shortcoming by focusing on the summarization process and developing techniques to trade-off map transmission size and computational performance.

Chapter 7

EFFICIENT MAP SUMMARIZATION

To better achieve scalability to large teams of robots, specifically in those cases where communication bandwidth is at a higher premium than computation, I developed summarization techniques that dramatically reduce the message sizes from the large dense summarized maps used in previous chapters. While the previously developed summarization techniques starting either from a local nonlinear graph (as in DDF-SAM 1.0) or from a Bayes tree in incremental approach (as in DDF-SAM 2.0), provide a basis for exact summarization, I have examined alternate approaches as a means to improve online performance and reduce the transmission cost of sending a summarized map between platforms. I classify these alternate techniques into *exact* summarization, such that the summarized density is equivalent to summarization performed in batch, and *approximate* summarization that is a conservative, but sparse representation of the joint over the shared variables.

These alternative techniques provide a means for DDF-SAM to scale to large teams of robots by making trade-offs between computational performance, message size and map precision. In particular, I will examine how to reduce the size of messages between robots at the expense of map precision through approximate summarization techniques, which can benefit robot systems where communication bandwidth is a tighter performance constraint than onboard computation.

Approximate techniques will also need to have extended requirements for ensuring consistency, in which not only do I need to avoid double-counting information, I also need ensure that each approximation summarization is conservative. This chapter will also examine extend these requirements with additional evaluation metrics for

DDF-SAM summarizations, and provide evaluations on both real-world and simulated datasets.

7.1 Exact Summarization

In this section, I show that the cost of transmitting summarized maps grows too fast to allow for truly scalable decentralized mapping. When performing summarization in an incremental context, as in the case of DDF-SAM 2.0, there are two options explored in previous chapters computing a consistent summarized map:

- Dense summarization from pure local factor graph: this technique is the same as in DDF-SAM 1.0 (see Algorithms 3 and 4), which ignores the Bayes tree and performs marginalization to compute a joint factor.
- Dense summarization from incremental Bayes tree: this technique was introduced with DDF-SAM 2.0 (see Algorithm 17), in which I liquefy a Bayes tree $\mathcal{B}^{N'_r}$with double-counted information using the dense summarization, and then apply antifactors to subtract the double-counted information.

In each of these cases, the primary computational cost is computing the joint $p\left(\Theta_s^r\right)$ from either a local factor graph Φ^r, which grows continuously with time, or from a Bayes tree $\mathcal{B}^{N'_r}$ with neighborhood information, which grows with communication with other robots.

7.1.1 Schur Complement Reordering

One possible alternative approach, Schur complement reordering, can yield an exact summarized map that is cheap to compute at summarization, but I will show the trade-off in computational performance during other update steps makes this approach ineffective in practice. This alternative strategy for summarizing from a Bayes tree is to explicitly structure the Bayes tree during incremental updates to

facilitate efficient summarization through choice of elimination ordering. In a typical case, the elimination ordering chosen during reordering stages in an incremental solver are computed to minimize the fill-in during elimination, which results in a sparse Bayes tree structure. However, is is possible to constrain this ordering to meet additional requirements, which I can exploit to make extracting a summarized map computationally easier.

The most straightforward approach is *Schur complement summarization*, in which as new variables are added to the Bayes tree incrementally, I constrain the ordering to place all of the shared variables at the base of the tree. I name this approach for the Schur complement trick frequently used in visual bundle adjustment work [90, 4], which creates a reduced graph over a subset of variables to perform more efficient solving. By placing all the shared variables at the base of the Bayes tree, extracting a summarized map from the Bayes tree becomes trivial. In this case, I move the computational cost of computing a large dense joint density over shared variables to a periodic reordering step of the full Bayes tree.

While this Schur complement trick does allow computationally cheaper summarization operations, the cost is in more expensive updates (both local and neighborhood), because this ordering yields a much denser Bayes tree than fill-reducing orderings. Particularly in the case of an iSAM 2.0 incremental solver (see Algorithm 22 for an overview, and [85] for details), which performs partial reordering during updates rather than reordering the entire system periodically, it is hard to enforce hard constraints on the position of variables in the Bayes tree. I evaluated this approach in the context of DDF-SAM 2.0 [33], and found that this trade-off did have any benefit over standard Bayes tree summarization (as in Algorithm 17), and as such are not including it in results presented in this document.

As a possible future work direction for this approach might be to adjust the position of variables within the Bayes tree through soft ordering constraints, in which

the ordering is computed to simultaneously minimize fill-in and keep shared variables near the end of the ordering. This line of research could allow for faster summarization with minimum additional computational cost and without needing a periodic full reordering.

7.1.2 Costs of Dense Summarized Maps

These exact summarization techniques have a key flaw for online performance in that they typically yield a dense Gaussian density $p\left(\Theta_s^r\right) \propto \mathcal{N}\left(\Lambda, \eta\right)$, in which the information matrix Λ grows *quadratically* with the dimension of the shared variables Θ_s^r. This growth in size of the summarized map messages will eventually become intractable as the robots explore, and motivates using approximate techniques as a means keep the communication tractable.

7.2 Approximate Summarization

To alleviate the problem of quadratic growth in message size, I introduce the approximate summarization, and propose additional necessary metrics for evaluating approximate map summarizations. A drawback to the exact techniques previously examined summarization is that the summarized joint $p\left(\Theta_s^r\right)$ over the saved variables is typically a fully connected distribution when expressed graphically, resulting in a need to send a dense matrix that grows quadratically with the dimension of Θ_s^r. This dense distribution impacts the size of messages transmitted between robots. In an ideal case, I would like to transmit a *sparse approximate* distribution $\tilde{p}\left(\Theta_s^r\right)$ that can be factorized over graph with a sparser structure. By enforcing a sparser structure, I can reduce growth in size of the messages between robots over time.

In single-robot SLAM cases, active sparsification has been an active topic of research, as it allows for managing a larger-scale and longer-term map. There are a variety of techniques for active sparsification in SLAM systems, such as sparse extended information filters (SEIF) [139], exactly sparse extended information filters

(ESEIF) [144], conservative sparsification (CS) [142] and generic linear constraint node removal [21, 22]. A detailed analysis of for conservative sparsification in SLAM can be found in [142]. [53] shows that the standard SEIF sparsification approach leads to an inconsistent absolute map, but relative geometry is consistent, and further presents a modification that fixes this problem at the cost of computational performance. The final result indicates that both the original and modified SEIF are overconfident.

Before presenting the approximate technique used for this approach, I first must examine how to quantify the *quality* and *conservativeness* of an approximation. To score quality of the approximation, in term how close it is to the true exact distribution, I use relative entropy, a.k.a., Kullback-Leibler divergence. Because it is possible to approximate a distribution in such a way that it can become overconfident in some dimensions, I apply a constraint to ensure conservativeness.

7.2.1 Quantifying Approximation Quality

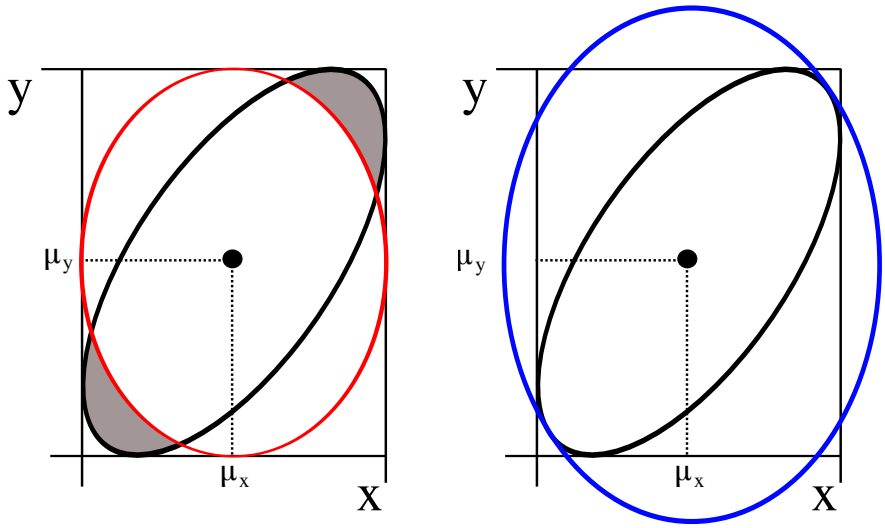
This section details the formulation the approximation quality metric, which yields a scalar score denoting the distance between reference exact distribution and an approximate distribution. In computing these scores for summarized maps, I use the local dense exact summarization (see 3.2) as our reference distribution. Note that these metrics are not intended to computed online, as they require operations that are too expensive to be tractable for larger systems.

The Kullback-Leibler divergence, also known as the relative entropy, as formulated by MacKay [97], for distributions $P(x)$ and $Q(x)$ over the same discrete variable set X is

$$D_{KL}(P \parallel Q) \triangleq \sum_{x} P(x) \log \frac{P(x)}{Q(x)}. \tag{32}$$

The key relationship is Gibb's inequality

$$D_{KL}(P \parallel Q) \geq 0 \tag{33}$$



(a) Naive Bayes (b) Conservative approximation

Figure 42: Example of the non-conservativeness of a naive Bayes approximation, in which the true distribution $p(x, y)$ is given by the black covariance ellipse. The naive Bayes approximation, shown on the left, is the red ellipse, constructed from the individual variances of $p(x)$ and $p(y)$ as projected onto their respective axes, which is non-conservative due to the area of the original ellipse not covered in the new approximation. The conservative approximation on the right maintains the independence of $p(x)$ and $p(y)$, but is inflated to ensure that the entire original ellipse.

which is equal iff $P = Q$. Note that the relative entropy is not a true distance, as it is not symmetric. For the particular case of relative entropy with coincident means with Gaussian distributions, the formulation is as follows, from [142],

$$D_{KL}(P \parallel Q) \triangleq \frac{1}{2}\left(\ln\left(\frac{\det\left(\Sigma_Q^{-1}\right)}{\det\left(\Sigma_P^{-1}\right)}\right) + tr\left(\Sigma_Q \Sigma_P^{-1}\right) - n\right), \tag{34}$$

where $P \triangleq \mathcal{N}(\mu, \Sigma_P)$ is the reference distribution, and $Q \triangleq \mathcal{N}(\mu, \Sigma_Q)$ is the approximate distribution of interest.

For approximate map summarization in DDF-SAM, the is to simultaneously minimize the KL divergence of approximate summarizations, while enforcing a sparsity structure that yields small between-robot message sizes and efficient updates.

7.2.2 Approximation Conservativeness

The other requirement on approximate summarization techniques is ensuring that the approximate summarization density Q is a *conservative* approximation of the true density P, such that approximation never has the effect of making an estimate more certain the true density. Ensuring that any approximations are conservative is necessary for maintaining consistency in DDF-SAM, in addition to the previous constraint of avoiding information double-counting. The standard formulation for conservativeness [14, 80] concerns the covariance matrices, where

$$\Sigma_Q \succeq \Sigma_P \tag{35}$$

which can be graphically interpreted through covariance ellipses, where the covariance ellipse of a conservative approximation Q will completely contain the true covariance ellipse of the true density P.

It is important to note that it is possible for an approximation both carry less information while still being non-conservative, as illustrated by in Figure 42, which shows graphically how a naive Bayes approximation is non-conservative. In this case, attempting to approximate $p(x, y) \approx p(x)\, p(y)$ does not account for the cross-correlations between x and y, yielding a density that is both approximate and non-conservative.

To account for non-conservative approximation, I can apply a correction to the underlying estimate that inflates the covariance ellipse (as in Figure 42b) such that the approximate density is both sparse (no additional correlation between variables) and conservative, at the expense of an approximation with greater divergence from the true density. This has been formally approached through covariance intersection techniques for data fusion [80, 79], which corrects for unknown correlations between variables by casting information fusion as a convex combination of factors. In a typical formulation fusing factors Φ in information form, where each factor $\phi_j \triangleq \mathcal{N}(\Lambda_j, \eta_j)$

with known correlations, information fusion corresponds to

$$\Lambda_{result} = \sum_{j \in \Phi} \Lambda_j,$$

I add a scaling factor on factors $\omega_j \in \Omega$ such that the result is

$$\tilde{\Lambda}_{result} = \sum_{j \in \Phi} \omega_j \Lambda_j$$

under the constraint that

$$\sum_{j \in \Phi} \omega_j = 1.$$

Ideally, the choice of scaling factor is the result of optimizing Ω to minimize the uncertainty added to the resulting estimate, as measured by the determinant of resulting covariance matrix, but for the purposes of this work, I will use a simpler uniform weighting scheme for scaling factors. In this case, the scaling factor $\omega_j \triangleq \frac{1}{N}$, where N is the number of factors being fused. I leave additional optimization of this correction for future work.

7.3 Naive Bayes Approximate Summarization

The first approximate summarization technique I introduce is naive Bayes, which provides the smallest and sparsest possible message size, at the expense of approximation quality, where underlying approximation is $p\left(\Theta_s^r\right) \approx \prod_{\theta_j^r \in \Theta_s^r} p\left(\theta_j^r\right)$. The resulting density optimally sparse, as there are no cross-correlations between variables, which results in both small message sizes as well as faster updates during information fusion, as there is no additional fill-in during elimination.

The default approach for computing variable marginals in a graphical system is to linearize and reorder, such that the target variable θ_j is at the end of the ordering, and then eliminate the entire system. This is an expensive operation for a single marginal, as it amounts to the same computational process as batch summarization, but must also be repeated for each variable. However, I can exploit the structure of

Algorithm 26 Naive Bayes summarization starting from a neighborhood Bayes tree $\mathcal{B}^{N'_r}$, incorporating both antifactor corrections for double-counted information, and a uniform covariance intersection correction for consistency.

`function summarizeNaiveBayes`$\left(\mathcal{B}^{N'_r}, \Theta^r_s, \mathcal{M}^r\right)$

  `initialize linear graph` $\tilde{\Phi}^r_{summarized} \leftarrow \emptyset$
  `for each variable` θ^r_i `in` Θ^r_s
    `compute marginal factor` $\tilde{\phi}_i \propto p\left(\theta^r_i\right)$ `from` $\mathcal{B}^{N'_r}$
    `add` $\tilde{\phi}_i$ `to graph` $\tilde{\Phi}^r_{summarized}$
  `for each` $\boldsymbol{m}^a_j \in \mathcal{M}^r$
    $\tilde{\Phi}^{-a}_j \leftarrow subtractDoubleCounted\left(\boldsymbol{m}^a_j, \Theta^r_s\right)$
    `add antifactors` $\tilde{\Phi}^{-a}_j$ `to` $\tilde{\Phi}^r_{summarized}$
  `scale factors` $\tilde{\Phi}^r_{summarized}$ `by CI weight` $\omega = {}^1/_{|\Theta^r_s|}$
  $\tilde{\Phi}^r_{summarized} \leftarrow condenseFactors\left(\tilde{\Phi}^r_{summarized}\right)$
  $\boldsymbol{m}^r_k \leftarrow createSummarizedMap\left(\tilde{\Phi}^r_{summarized}, \Theta^r_s, k\right)$
  `return` $\boldsymbol{m}^r_k$

---

the Bayes tree to cache the separator marginals, which allows us to only eliminate small portions of the tree at a time.

The overall algorithm, as in Algorithm 26, is straightforward. Because this approach starts from a DDF-SAM 2.0 Bayes tree that combines both local and neighborhood information, as well as being an approximate density, there are two correction stages applied after constructing the underlying summarized graph: antifactors to subtract double-counted information and uniform covariance intersection scaling. Naive Bayes summarization is particularly amenable to factor condensing after the addition of antifactors, as the antifactors (assuming all robots are using the same summarization technique) will also be disconnected. These factors can condense efficiently to a single constant-size information-form factor each variable in the system.

As a result of this summarization approach, there are two key benefits

- The transmission size of a summarized map grows *linearly* with the number of saved variables.

- Fusing a completely disconnected summarized map on a receiving robot should be cheaper.

Because there is only one factor per shared variable, and the size of these factors are determined only by the dimension of the shared variable, it follows that message size growth rate will be linear. This is in contrast to the quadratic message size growth rate of exact approaches, which grows quadratically with the number of shared variables due to sending cross-correlations between all variables. Because the summarized graph is disconnected, fusing with another robot's local system will be cheap because the additional factors will not increase the density of the system at all, resulting the fastest possible update.

While reduced communication cost and neighborhood update time are a significant, this approach has two primary drawbacks:

- Low approximation quality due to removing all of the correlations between variables.

- Disconnected shared variables that are do not overlap with the receiving local map will not contribute to information fusion.

Between these shortcomings, the most significant is the effect of representing a summarized map as a disconnected graph, as those variables that would extend the sensor horizon of a receiving robot do not play a role in the local fusion problem, and will remain uncertain until they are observed locally by the receiving robot. Particularly due to the inflation of covariance ellipses with a uniform CI correction, the extended sensor horizon will have much more uncertain estimates than a representation with a connected graph.

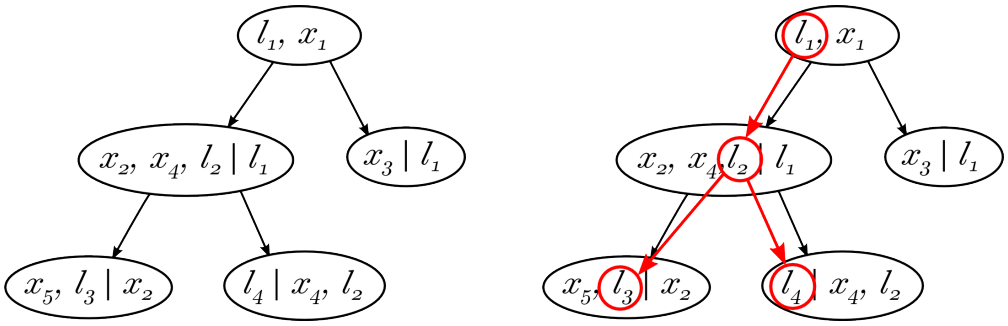


Figure 43: An example first order tree summarization, starting from an existing Bayes tree (left) that incorporates both poses and landmarks, and the nearest-shared-parent tree structure overlaid in red (right).

7.4 Local Tree Approximate Summarization

In this section, I introduce a tree approximation which maintains the linear message size growth of the naive Bayes approximation, while achieving substantially improved approximation quality and improved map fusion. To move beyond the naive Bayes approximation, the clearest option is to use a first-order tree to perform summarization, as illustrated in Figure 43, which will remain linear in transmission size with the number of variables, while being a better approximation of the true joint density $p\left(\Theta_s^r\right)$. A first order tree is one in which each variable is conditioned on at most one other variable. This tree structure is also an improvement on the naive Bayes approach in that it is a connected graph, which allows for the uncertainty in non-overlapping areas of the to be reduced through connections to the local graph.

I will use a tree approximation designed to be efficient to compute from a Bayes tree structure, to allow for faster summarization, and leave optimal approximation for future work. The optimal, in terms of KL divergence, first-order tree is the Chow-Liu tree (CLT) [31], which is created by computing a maximum spanning tree over a mutual-information graph over the true distribution. See [29] for a more extensive application of the CLT to SLAM applications.

The local tree summarization algorithm I introduce in this section is drastically different from the previous summarization approaches for DDF-SAM 2.0 presented in

Algorithm 27 Tree summarization from local factor graph Φ^r.

```
function summarizationTreeApprox(Φ^r, Θ^r_s)
    Φ̃^r ← linearize(Φ^r, Θ^r_s)
    B^r ← multifrontalEliminate(Φ̃^r)
    Φ̃^r_summarized ← ∅
    for each variable θ_i in Θ^r_s
        if θ_i is root-most of Θ^r_s in B^r
            compute φ̃_i ∝ p(θ_i) from B^r
            add φ̃_i to graph Φ̃^r_summarized
        else
            find closest parent θ_j in Θ^r_s in B^r
            compute φ̃_ij ∝ p(θ_i|θ_j) from B^r
            add φ̃_ij to graph Φ̃^r_summarized
    scale factors Φ̃^r_summarized by CI weight ω = 1/N
    m^r_k ← createSummarizedMap(Φ̃^r_summarized, Θ^r_s, k)
    return m^r_k
```

this thesis, as it abandons the antifactor mechanism for computing summarizations from an incrementally-updated Bayes tree. Below, I introduce the general structure of the algorithm, and then provide insight into design decisions behind choosing to use the pure local graph for summarization instead of the neighborhood Bayes tree, as well as the choice of the tree use for approximation.

The general algorithm for creating a tree-structured summarized map from local factor graph is presented in Algorithm 27, which computes the first-order tree using a nearest-shared-parent tree structure. Figure 43 illustrates this tree structure overlaid on a given Bayes tree. The algorithm starts by first linearizing and eliminating the full local system, which is a computationally expensive operation, though the linearization step can be short-circuited by using previously cached linear factors rather than linearizing directly. Because this is an approximate summarization technique, it requires the same uniform CI correction as in the naive Bayes case to ensure that the resulting approximation is conservative.

This summarization technique departs from the previous DDF-SAM 2.0 techniques in that I start from the pure local system rather than the neighborhood Bayes tree, as the necessary antifactor corrections do not work under approximation outside of the naive Bayes case. Because the antifactor downdating step does guarantee correctness under approximation, it is not possible to undo contributions made from neighboring robot to avoid information double-counting. Normally, I summarize and then apply exact antifactors derived from cached summarized maps to undo these contributions. However, applying exact antifactors will fail if the summarization step is itself approximate, because, unlike with an exact summarization, the contributions to the neighborhood Bayes tree from the summarized maps have been approximated by the summarization operation, and therefore are no longer equivalent to antifactors constructed from the summarized maps directly. Because I cannot reliably perform antifactor corrections, it is not feasible to start with a neighborhood Bayes tree.

The reason that antifactors fail in the case of this approximate summarization approach is that it is not possible to guarantee that antifactor downdating will yield a summarized map that is either ill-conditioned or including double-counted information. To illustrate this point more formally, I will represent the system in terms of adding contributions in information form using information matrices, where an exact summarization on a Bayes tree $p\left(\Theta_s^r; \mathcal{B}^{N_r'}\right) \propto \mathcal{N}\left(\Lambda_{total}\right)$ before adding antifactors can be represented as

$$\Lambda_{total} = \Lambda_{local} + \sum_{j \in \mathcal{M}^r} \Lambda_j, \tag{36}$$

in which the goal for antifactor downdating is to solve for Λ_{local} by subtracting contributions from other robots Λ_j. This works because each summarized map $\boldsymbol{m}_j$ yields the exact contribution Λ_j, so Λ_{local} can be found exactly. In the case of an approximate summarization $\tilde{p}\left(\Theta_s^r; \mathcal{B}^{N_r'}\right) \propto \mathcal{N}\left(\tilde{\Lambda}_{total}\right)$, the same relationship between information contributions still holds, as in

$$\tilde{\Lambda}_{total} = \tilde{\Lambda}_{local} + \sum_{j \in \mathcal{M}^r} \tilde{\Lambda}_j, \tag{37}$$

such that there is an target approximate local density $\tilde{\Lambda}_{local} \triangleq \tilde{\Lambda}_{total} - \sum_{j \in \mathcal{M}^r} \tilde{\Lambda}_j$, and contributions from summarized maps, which have each been approximated in the computation of $\tilde{\Lambda}_{total}$. The challenge for arbitrary approximations is determining the contributions from the other robots under approximation$\tilde{\Lambda}_j$, which can vary with the sparsity pattern imposed by the approximation technique. If I compute a contribution $\tilde{\Lambda}'_j$ such that it is *more* certain than the true contribution $\tilde{\Lambda}_j$, I subtract too much information from $\tilde{\Lambda}_{total}$, yielding a summarized map that may be non-positive definite - effectively a summarized map that subtracts information on the receiving robot. If I choose $\tilde{\Lambda}'_j$ such that it is *less* certain than $\tilde{\Lambda}_j$, there will be double-counted information remaining in $\tilde{\Lambda}_{local}$, resulting in overconfident estimates. The naive Bayes approximation is a special case in which $\tilde{\Lambda}'_j$ can be reliably predicted from $\tilde{\Lambda}_j$ due to the completely disconnected structure.

The *nearest shared parent* tree structure described in Algorithm 27 uses a simple heuristic to match a first-order tree to the Bayes tree structure from elimination: each shared variable θ_i is conditioned on the closest parent shared variable θ_j. This structure is designed to exploit the caching of separator marginals in the Bayes tree to speed up summarization. Furthermore, this structure results in a first-order tree that provides reasonable approximation of the structure of the Bayes tree itself, yielding a simple heuristic for sub-optimal, but effective approximations.

The overall benefits of the local tree approximate summarization are

- The transmission size of a summarized map grows *linearly* with the number of variables, and different than naive Bayes by a constant factor.
- A tree approximation will carry more information than the naive Bayes approximation.

- Making the summarized map a connected graph ensures that variables in the extended sensor horizon benefit from fusion with the local map.

The most significant of these advantages over naive Bayes is the graph connectivity through which an approximate summarization, even with a substantially inflated covariances, can yield a more certain extended sensor horizon region of the map.

The primary disadvantage of this approach the additional computation required due to re-eliminating the full local system, rather than starting from an existing Bayes tree, which may mean that this approach is better used in robot applications where communication is more constrained than computation. In practice, it may be possible to alter this scheme to incrementally maintain a separate Bayes tree that is purely local for summarization purposes, and thereby mitigate the computational cost at summarization.

7.5 Evaluation

I evaluate these summarization techniques simulated datasets as in the previous chapter (see Section 6.4 for results, and Section3.4 for details on the evaluation benchmarks), as well as using a real-world dataset as a performance benchmark for summarization, evaluated with the more thorough consistency metrics described in this chapter. These experiments address the following core DDF-SAM claims:

- Online: I evaluate online operation concerns through timing performance of summarization and update operations in real-world and simulated datasets.

- Scalable: I measure the size of summarized maps as proxy for network bandwidth, which validate claims that the system is scalable to large teams of robots with regard to network bandwidth.

- Consistent: Directly evaluating the conservativeness of summarization techniques, particularly focusing on approximate techniques.

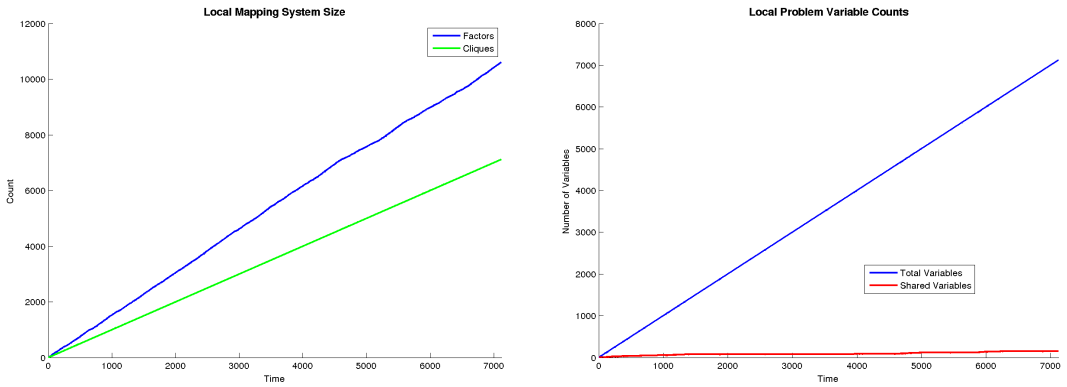


Figure 44: Dataset size statistics for a single-robot run through the Victoria Park dataset, with factor/clique counts over time (left) and the variable count over the course of the dataset (right). Note that the number of shared variables grows at a much smaller rate than the total number of variables in the system.

- Extended sensor horizon: I evaluate the impact of approximate summarization techniques on the resulting nonlinear DDF-SAM 2.0 solutions to demonstrate that approximate summarization techniques produce an extended sensor horizon.

The primary addition in evaluation scenarios a single-robot real-world dataset, designed to to act as a benchmark for summarization decoupled from neighborhood map fusion to validate both performance and consistency claims. For simulations, I use the Manhattan-world scenario for qualitative analysis of the map estimates, and the randomly generated large scale scenario to compute aggregate statistics.

The real-world dataset employed is the standard Victoria Park dataset [65] to benchmark summarization performance, approximation quality, and transmission size as a single-robot dataset. In this case, the benchmark performs summarization of the local system at fixed intervals, which provides a baseline performance estimate for summarization when isolated from a true multi-robot system. As a point of reference for the size of the dataset and corresponding summarized maps over time, Figure 44 shows statistics for the graph size in an incremental environment, as well as the relative growth of total variables and shared variables.

7.6 Results

I completed the experiments described in the previous section and present the results, which validate the following claims:

- Online: The approximate summarization techniques come with the trade-off of higher performance costs, which grow faster than exact techniques, but remain tractable.
- Scalable: The transmission size of summarized maps using sparse approximations match the expected linear growth with number of variable size, which yields a dramatic reduction in communication costs.
- Consistent: These results show that the local tree approximation combined with CI correction is a conservative approximation, yielding consistent DDF-SAM map estimates, and all of the techniques yield reasonable estimation error results.
- Extended sensor horizon: While the approximate summarized maps yield greater uncertainty in map estimates, the augmented local maps maintain an effective extended sensor horizon.

I present the results, starting with the Manhattan-world scenario to qualitatively demonstrate the effect of approximate map summarization, followed by the Victoria Park summarization benchmark, and then the large scale multi-robot simulation.

7.6.1 Manhattan-world Simulation

The Manhattan-world scenario results, with controls of a pure local solution and the exact dense summarization in a nonlinear DDF-SAM 2.0 system shown in Figure 45, provides a clear illustration of the effect of using approximate summarization techniques, which are shown in Figure 47. Of particular interest in these figures is

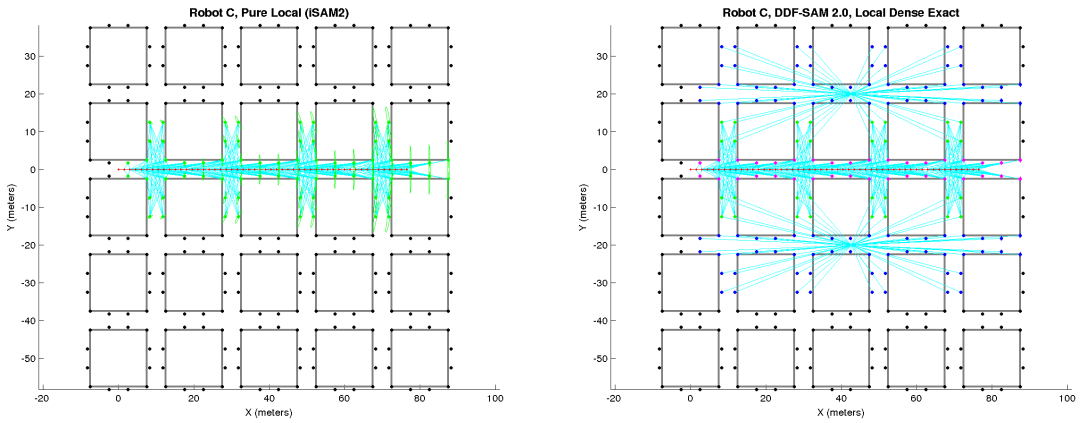


Figure 45: Map solutions with landmark marginals drawn for Manhattan-world simulated dataset for a single robot, showing a purely local solution (left) and the result of nonlinear DDF-SAM 2.0 with two fusion neighbors. Landmarks in the multi-robot example are color-coded in groups for pure local, overlapping, and pure neighborhood variables, as magenta, green and blue, respectively. Both renderings include covariance ellipses for all landmarks in the system.

the relationship between the size of covariance ellipses for the pure neighborhood information (colored in blue) and graph connectivity of those regions.

In the extended sensor horizon area of the map, the tree approximation provides much tighter uncertainty estimates than in the case of the naive Bayes approximation, which can be attributed in large part to the additional graph connectivity between the neighborhood contribution of the summarized map and the local map. The actual summarized maps, rendered with covariance ellipses are shown in Figure 46, highlight this difference, as it is the naive Bayes summarization with tighter ellipses in this comparison. The reason for the discrepancy is that while a naive Bayes summarized map may be more precise by itself, when fusing with the local map of a robot, as in Figure 47, only variables overlap directly are affected by information fusion. In the local tree summarization case, fusing with the local map results in an improved estimate for even those variables that do not overlap.

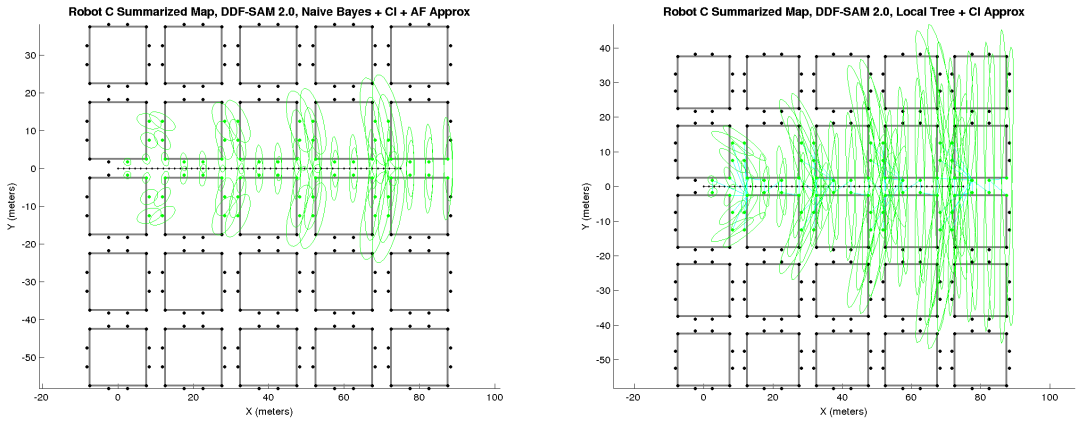


Figure 46: Approximate map summarizations using naive Bayes (left) and local tree approximation (right) for a single robot in the Manhattan-world scenario with covariance ellipses (green) and factor lines drawn (cyan).

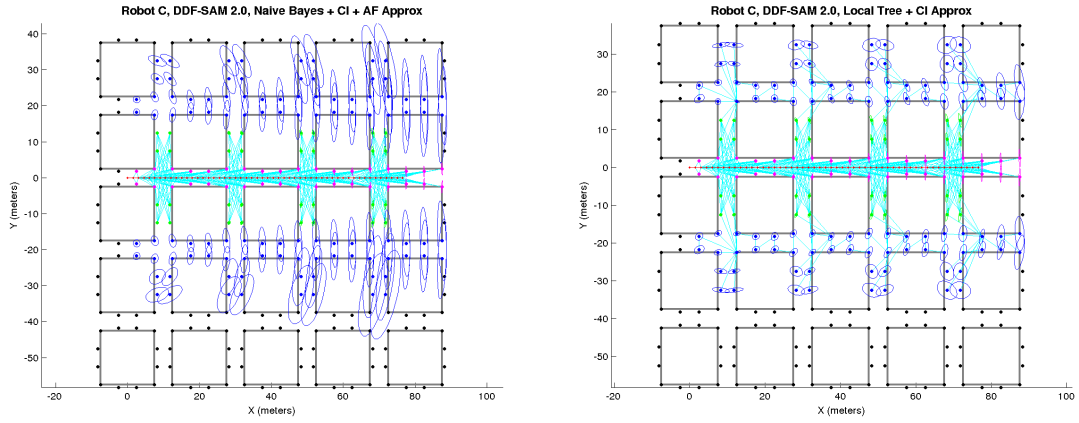


Figure 47: Map solutions with landmark marginals drawn for a single robot in the Manhattan-world dataset for approximate summarization techniques, showing naive Bayes (left) and local tree approximate summarization. These renderings use the same color coding as in Figure 45.

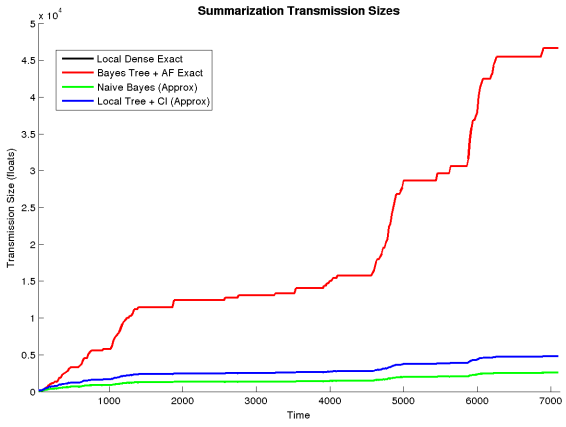


Figure 48: Summarized map transmission size over time for each summarization type using the Victoria Park dataset as a single-robot scenario. Note that this does not include any size impact of antifactors because this is only a single-robot application. Both the local dense summarization and Bayes tree exact summarization have the same transmission size, and have overlapping lines as a result.

7.6.2 Victoria Park Single-robot Summarization

The Victoria Park dataset, which is large enough to show statistical trends over time, highlights the core improvements that the sparse approximate summarization techniques have over exact techniques in terms of map transmission size, as plotted in Figure 48. The transmission size for both approximate summarization techniques remain flat over time, even when the exact summarization techniques increase significantly in size due to additional shared variables. Between the two approximate summarizations, the local tree approximation is roughly twice the transmission size of the naive Bayes approximation, which is as expected from the theory.

While local tree approximation has a larger transmission size by a constant factor, it can also be seen that this additional message space provides a large increase approximation quality. As shown in Figure 49, the divergence between an exact summarized map and the naive Bayes summarization grows rapidly, whereas that of the local tree approximation does not.

In addition a less certain solution, the approximate summarization techniques have the additional cost of longer compute times, as shown in Figure 50, where both

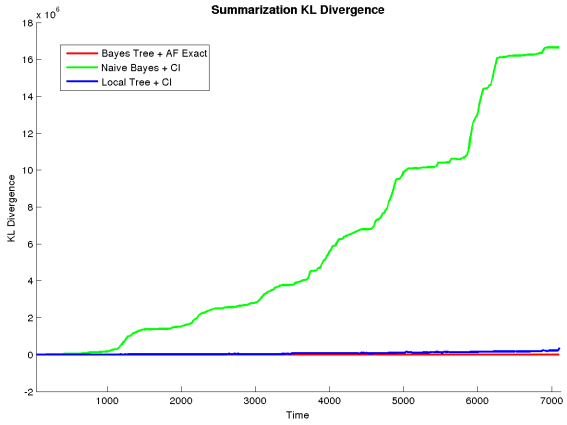


Figure 49: Kullback-Leibler divergence for summarization techniques in a single-robot benchmark scenario of the Victoria Park dataset, using dense local summarization as the ground truth distribution.

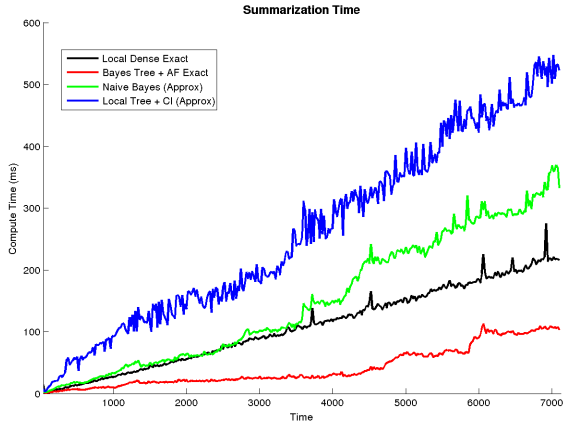


Figure 50: Summarization timing of different summarization techniques on Victoria Park dataset.

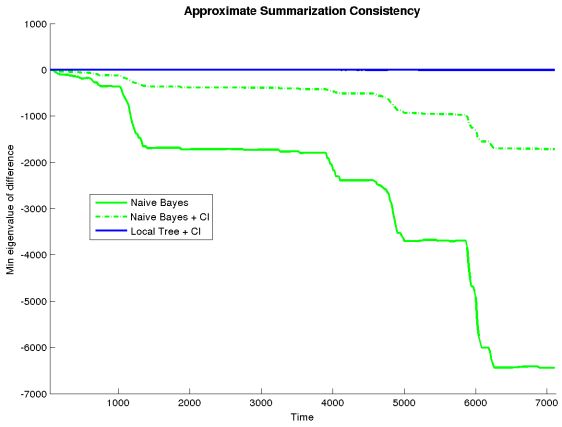


Figure 51: Summarization conservativeness for approximate summarization techniques on the Victoria Park dataset, where the scoring function is the smallest eigenvalue of $\Sigma_Q - \Sigma_P$, which should be positive to be strictly consistent. Also plotted, for comparison, is the consistency score for naive Bayes summarization without the CI correction, which in this case is clearly not consistent.

the naive Bayes and local tree approximations have higher computational costs that grow at faster rates than either exact summarization technique. It is not surprising that local tree approximation yields the highest computational cost, as it must first re-eliminate the entire local system, which is the equivalent operation as local dense exact summarization, and then do additional computations for each edge in the tree. It should be noted that for both Bayes tree exact summarization and naive Bayes approximate summarization, the timings for a single-robot scenario will increase with the addition of summarized maps that need to be negated through anti-factors.

Consistency remains a challenging problem in the context of summarizations, as illustrated in Figure 51, which shows consistency scoring for the approximate summarizations, which highlight the effect applying a uniform CI correction. Local tree summarization is demonstrably conservative throughout, though naive Bayes, even with a CI correction, becomes non-conservative over time. The addition of a CI correction for naive Bayes does, however, reduce the degree to which the solution will become overconfident.

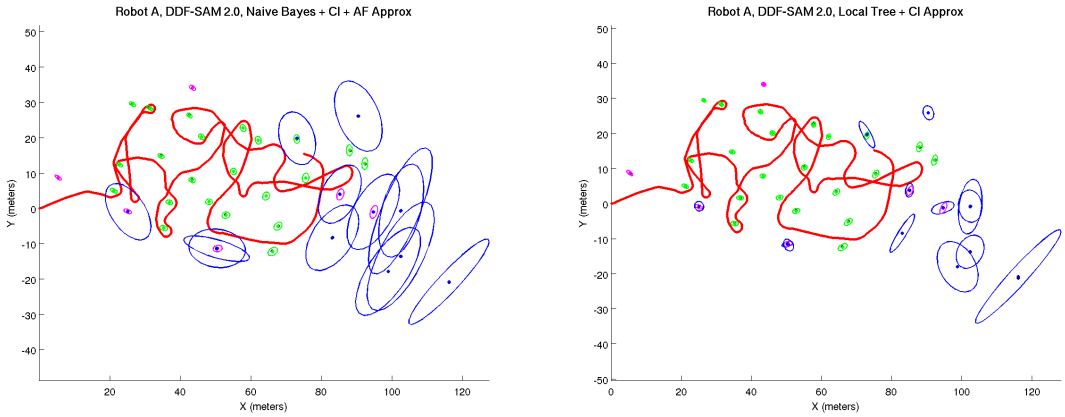


Figure 52: Renderings of solutions from Freiburg dataset run 2, showing the final solution for robot A as generated with naive Bayes (left) and local tree approximate summarization. The corresponding solutions for this robot using exact summarization are in Figure 31.

7.6.3 Freiburg Dataset

As in the Manhattan world example, the difference in the solution quality in the Freiburg dataset results can be visually observed through the difference in the covariance ellipses for the landmarks, shown in Figure 52. Between the two approximate summarization techniques, it is clear that naive Bayes results in much larger covariance ellipses when fused into a solution, particularly in comparison to the local tree approximate summarization technique. In comparison to the solutions generated using exact summarization, shown in Figure 31, the shared variables (colored green in the renderings) for the approximate techniques have somewhat more uncertainty than the exact case.

Figure 53 shows summarization costs, both the compute time for generating a summarized map and the transmission size for each summarized map under the approximate summarization techniques as compared to exact summarization. With regard to compute time for summarization, there are two clear clusters depending on whether it is necessary to fully re-eliminate the underlying local system at every summarization interval, in which local tree approximation tracks the higher linear

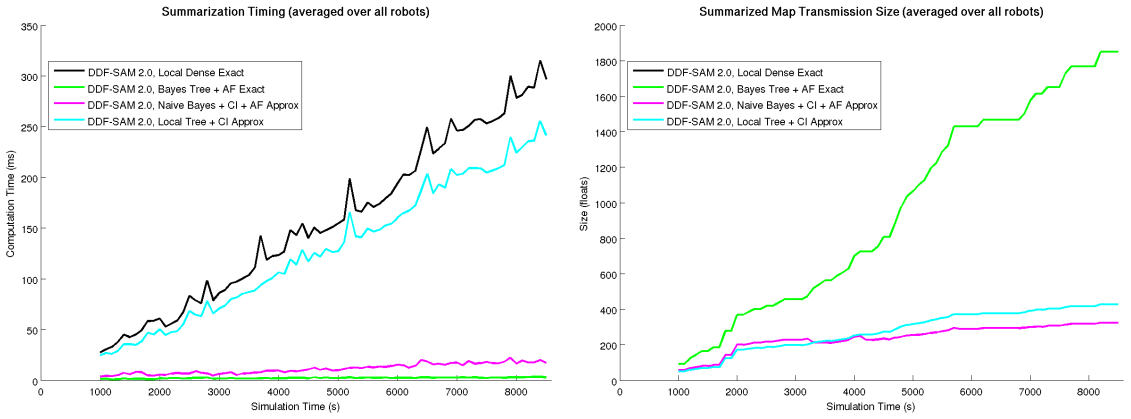


Figure 53: Summarization costs for nonlinear DDF-SAM 2.0 summarization techniques from Freiburg dataset run 2, showing the time to compute a summarized map (left) and the transmission size of summarized maps. Note that in the transmission size plot, the values for Bayes tree + antifactor exact summarization are identical to the local dense exact summarization.

cost of local dense exact summarization, and naive Bayes summarization tracks the much lower cost of summarization directly from the Bayes tree. Both Bayes tree exact and naive Bayes approximate summarization avoid the need to re-eliminate the entire system at the start of summarization, which yields summarization times that are faster and grow at a slower rate with time. As in the previous cases, with regard to summarized map transmission size, the approximate techniques grow at a much slower rate with time, which is as expected.

Between the different nonlinear DDF-SAM 2.0 summarization techniques, the update timing costs for adding either local measurements or incoming summarized maps do not vary significantly. This is as expected for this dataset because of the large number of local factors mean that any benefit that the approximate techniques show in the graphical density of the summarized map factors is insignificant compared to the local system.

Figure 54 shows the solution accuracy for each of the summarization techniques, in which it is clear that the approximate techniques converge at the end of the dataset to a higher absolute error than the the exact summarization techniques. This is the

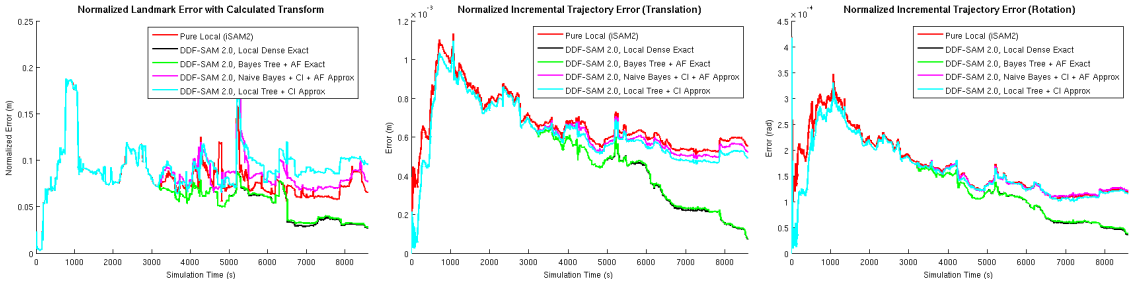


Figure 54: Error in the solutions in comparison to the centralized batch solution for the Freiburg dataset run 2, showing, from left to right, the landmark constellation error, the error in translation for the trajectory, and the error in rotation for the trajectory. These plots show nonlinear DDF-SAM 2.0 results for the two approximate summarization techniques compared to the exact summarization techniques.

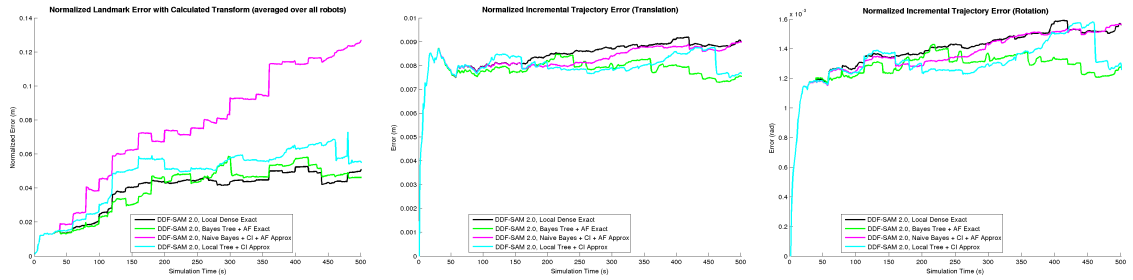


Figure 55: Effect of summarization techniques within the large scale simulation on solution error metrics (landmark constellation error, trajectory translation, trajectory rotation from left to right) for the large scale simulation with 25 robots and neighborhood bound $K = 4$, showing both exact summarization techniques, as well as naive Bayes and local tree approximations.

expected result as the introduction of approximation in summarized maps has the effect of weighting neighborhood information substantially less than locally available map information, resulting in the solution error converging closer to that of the pure local solution, rather than that of the centralized multi-robot solution.

7.6.4 Large Scale Simulation

In the full multi-robot scenario of the large scale simulation, I plot same metrics as in Section 6.5, which show the estimation error (solution error for landmarks and trajectories shown in Figure 55 and for relative reference frames in Figure 56) in the approximate cases is on par with the exact summarization techniques. The largest variation in error is in landmark constellation error for naive Bayes, which appears

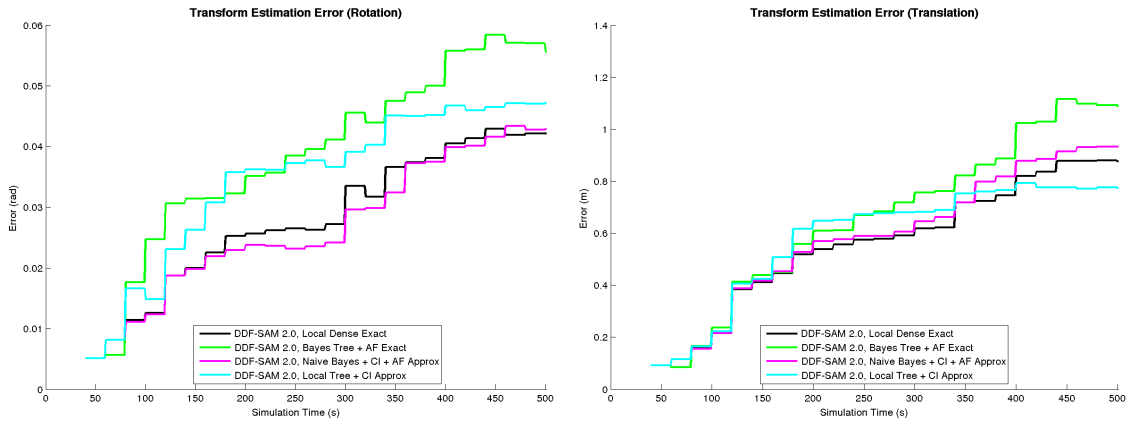


Figure 56: Estimation error for transforms between frames, split into rotation (left) and translation components with results for each nonlinear DDF-SAM 2.0 summarization technique shown.

to continue growing as the error in the other techniques stabilize.

Summarization performance, shown in Figure 57, follows the trends from the Victoria park dataset, where summarized map transmission sizes grow substantially slower than the exact techniques while showing the same summarization timing profile. Critically, the summarized map transmission size grows slower than the cost of sending raw nonlinear measurements to neighboring robots.

The evaluations of summarization quality, plotted in Figure 58, show a similar profile as well, with the naive Bayes performing poorly in approximation quality and local tree summarization remaining much closer to being exact. As with the Victoria park data, there is the problematic trend in summarization consistency of naive Bayes becoming nonconservative over time, a trend that it also shares with Bayes tree summarization. While the Bayes tree summarization does become nonconservative, it also appears to be nearly exact in terms of KL divergence.

When examining the effect on local and neighborhood updates, plotted in Figure 59, there is a small improvement in the local update timing for naive Bayes summarization, but the approximate summarization techniques have the unexpected result of having slower neighborhood updates.

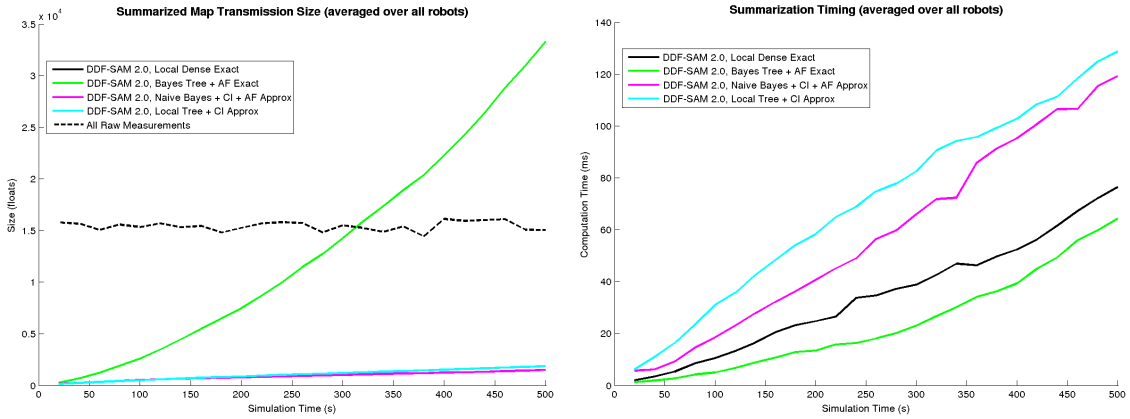


Figure 57: Effect of summarization techniques on summarized map transmission size (left) and compute time for summarizations for the large scale simulation with 25 robots and neighborhood bound $K = 4$, showing both exact summarization techniques, as well as naive Bayes and local tree approximations. Note that in the case of the summarization size, both local dense exact and Bayes tree + AF exact have the same transmission size.

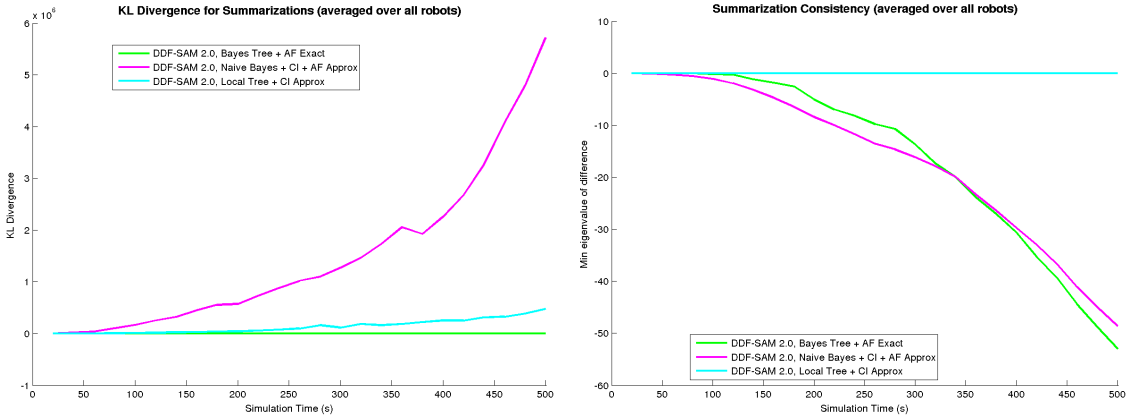


Figure 58: Effect of summarization techniques on approximation quality (left) and summarization consistency for the large scale simulation with 25 robots and neighborhood bound $K = 4$, showing Bayes tree summarization, as well as naive Bayes and local tree approximations.

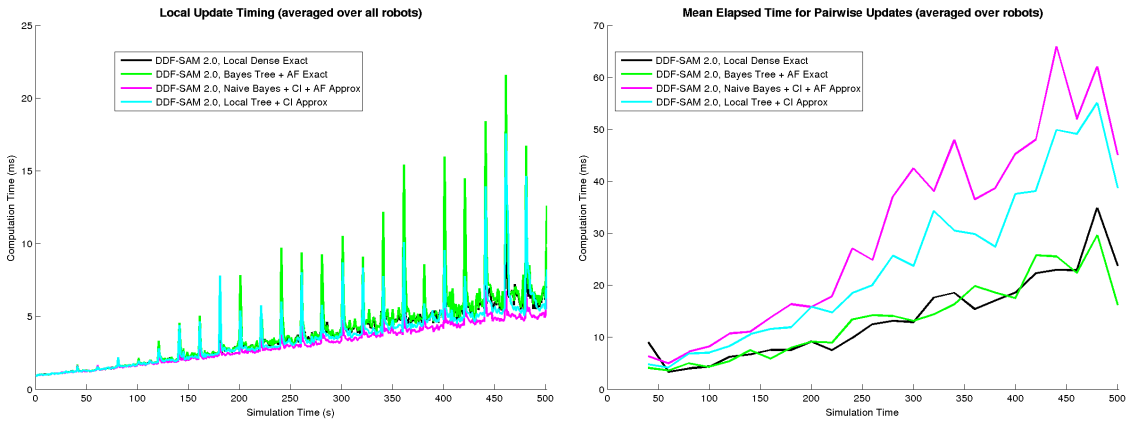


Figure 59: Effect of summarization techniques on update timing, both for local update timing (left) and for pairwise updates between neighboring robots for the large scale simulation with 25 robots and neighborhood bound $K = 4$.

7.7 Discussion

As an extension of the exact summarization techniques used in previous techniques, approximate summarizations provide a substantial improvement in the communication cost - going from quadratic to linear growth with the number of variables - at the cost of approximation and additional computation time. For robots with tighter constraints on communication than on computation, choosing approximate summarizations, particularly the local tree-structured approximation, yields a substantial benefit.

While the results presented with regard to using approximations for map summarization show that it is feasible to share a smaller amount of information between platforms and still yield a reasonable map result, this technique could be improved through future work to address the optimization for both approximation quality and conservativeness directly. The most direct route is to use the Chow-Liu tree, which optimal in KL divergence, and optimize over covariance scaling parameters to inflate only enough to ensure a conservative estimate, which would reduce the additional uncertainty added through approximation.

Chapter 8

DESIGN TRADE-OFFS

While the previous chapters provided a more detailed analysis of the DDF-SAM algorithms and results on particular datasets, in this chapter I will discuss more general design considerations for the use of DDF-SAM within real-world robot systems. Because DDF-SAM focuses on only the pure perception problem, which would be a subsystem within an autonomous or semi-autonomous multi-robot team, it is worth considering the implications of perception performance and design within a robot system. Much of this thesis has focused on a largely application-agnostic approach to solving multi-robot perception, there remain some fundamental assumptions that become significant when moving to a closed-loop multi-robot autonomous system.

8.1 Multi-robot SLAM in Context

A starting point for the design of a decentralized SLAM system is to examine the requirements for a multi-robot mapping system operating under varying degrees of autonomy, under different applications. Given these design requirements, it is possible to analyze the implications of performance metrics for the perception system on the full multi-robot system.

8.1.1 Team Objectives

The first consideration for multi-robot system is what sort of objective is required for successful completion of a mission. Without a specific objective, it is not possible to evaluate system performance, or drive requirements for the perception system. A straw-man example would be a team of robots that does not sense its environment, does not communicate, and does not move - this team would not be able to

application-specific goals such as searching an area for hazards or disaster survivors, nor would it be capable of accomplishing goals for most applications. I can characterize the types of system objectives that could make use of DDF-SAM as those in which the robots must interact in some way with their environment, in at least one of two ways:

- Physical motion: the robots actively move and navigate, such as an objective to explore all of a building where all of the robots start in some common location.
- Sensing: the robots need to acquire some knowledge of their environment, e.g., detect the presence of a disaster survivor in a building.

Note that it is possible for a team of robots to have a purely sensing objective, in which the team of robots acts as a static sensor network to measure or track elements in the environment. The typical case addressed by DDF-SAM is one in which the robots need to move through their environment. One could consider the case of a mission objective with a transition, in which one deploys a team of robots with the intention of creating a static sensor network, but the robots need to move from their starting location to find static locations in the environment. In the rest of this chapter, I will focus on the case of a robot team primarily requiring movement throughout the environment.

8.1.2 SLAM for Robot Teams

Given that the objective requires movement through the environment, intermediate requirements on the system become apparent. In cases where robots need to navigate, having knowledge of the environment can make this task easier. I will assume that the robots are navigating in an environment in which more reactive behavior-based robot navigation (as appears in a Roomba) will struggle due to hazards in the environment, as well as the an implicit requirement that the movement objective be carried out efficiently with time. A consistent metric map that can provide a basis for navigation

is both useful and necessary in these situations. Furthermore, a larger, more complete metric map can improve navigation performance by allowing for longer look-ahead windows during planning.

The intended requirements for a SLAM system that is beneficial to a multi-robot team in the applications described can be summarized as follows:

- The map should be beneficial for navigation and planning, such that navigation algorithms can efficiently and safely maneuver the robot.
- The map can be updated online as robots explore an environment.
- For applications targeting exploration of a given area, a good map should provide coverage of the space.
- Particularly in cases where there are hazards, the mapping system should be robust to robot failure.

In these cases, we want a decentralized perception problem able to provide mapping solution that satisfies this online navigation.

8.1.3 Evaluating DDF-SAM in Context

DDF-SAM approaches this mapping problem from a decentralized perspective, which targets a level of performance that is greater than that of individual robots performing pure local SLAM, with an upper bound of a fully centralized SLAM solution. DDF-SAM trades off map completeness for online performance by focusing on an extended sensor horizon for each robot, rather than trying to assemble a globally consistent metric map in centralized form.

The map solutions that DDF-SAM target the needs of a navigation algorithm, in which having a metric map of the local area of the robot is immediately beneficial because navigation can use a longer planning horizon with greater certainty. Because far away map information is less likely to be useful for a robot navigation algorithm

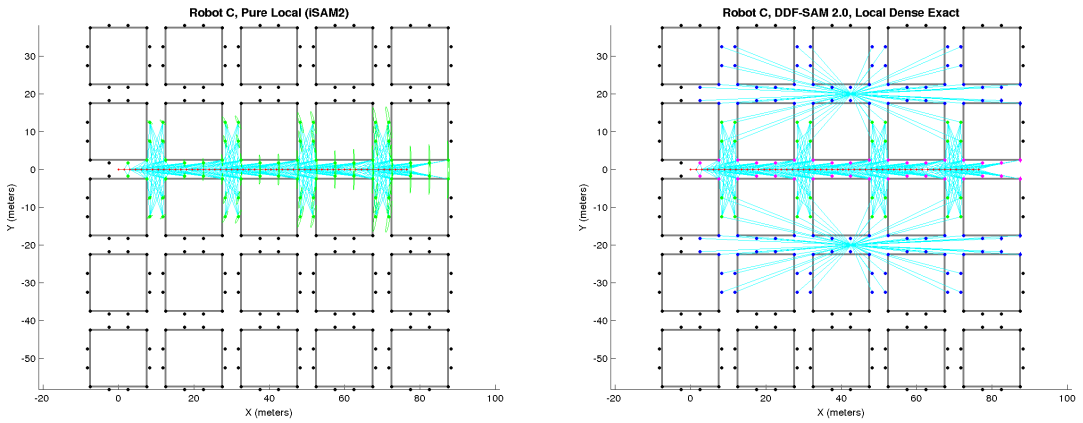


Figure 60: Map solutions with landmark marginals drawn for Manhattan-world simulated dataset for a single robot moving from left to right, showing a purely local solution (left) and the result of nonlinear DDF-SAM 2.0 with two fusion neighbors. Landmarks in the multi-robot example are color-coded in groups for pure local, overlapping, and pure neighborhood variables, as magenta, green and blue, respectively.

simply because the planning horizon is still limited, the loss of the global map from a centralized solution is not significant.

I evaluate DDF-SAM within this context of providing a navigation-aiding map solution near the robot. As a proxy for measuring the area covered, I can use the number of additional landmarks determine how useful a map might be to a given robot. As the clearest demonstration of the ability to increase the sensing horizon, and thereby increase the planning horizon of a robot, the Manhattan-world scenarios (shown in Figure 60) show that the area covered over the local solution increases substantially with the addition of neighborhood information.

The optimal planning strategy for robots to maximize their extended sensor horizons via DDF-SAM would need to minimize redundant area coverage, as in the Manhattan-world case. During the majority of experiments conducted in the evaluation of DDF-SAM, I did not constrain the trajectories of the robots for optimal team coverage as a scope limitation for the project. In the large scale simulation (see Section 3.4.6.2) and in the Freiburg dataset (see Section 3.4.6.3), the robots cover a great deal of overlapping area, so incoming summarized maps were frequently mostly

shared information. This redundant information is still useful, however, as it can improve the certainty of map solution for each of the robots.

The majority of the effort in this dissertation has come from managing the costs associated with decentralized inference at different steps, in order to demonstrate both online performance and scalability to large teams of robots. The primary performance measure for online operation is the *per-robot* compute time the local update, summarization and neighborhood updates. In addition, I am also concerned with communication bandwidth costs incurred by robots transmitting data to each other. The more complex metric is in evaluating the scalability of the the solution, which is concerned with both minimizing costs as well as minimizing the rate at which costs grow.

- Increasing team size: Adding a new robot r to a given robot team only directly impacts the costs for the communication neighbors N_r of r, where each robot $a \in N_r$ incurs an additional communication cost for sharing map information with r. However, the communication and computation cost for each robot a will only increase until it reaches the neighborhood bound K, resulting in costs that are constant with the number of robots.

- Environment size: For a scenario in which the robots are a bounded environment of size M (where M can denote the number of landmarks or the area covered) and robots actively explore during the mission duration, both the computation and communication costs will increase with M. If shared variables are chosen to be landmarks, the communication cost will reach an upper bound in which each robot has explored the entire area. This communication cost grow until reaching the bound as either $O(M^2)$ for exact summarization or $O(M)$ for approximate summarization.

- Mission duration: The compute costs will grow without bound as each new

timestep will add another pose variable and more landmark observations. The growth rate can be loosely characterized by the ratio between newly observed and re-observed variables, where re-observing variables adds to the graph density, but it will depend on the factor graph topology. When using an incremental solver, as in nonlinear DDF-SAM 2.0, this compute cost is roughly linear with time from the experiments conducted.

Because of the possible variations in robot exploration paths and the structure of the environment, it difficult to make hard claims regarding costs during operation. As a root challenge, it is difficult to make strong claims regarding the performance of optimization in the sparse graphical models that underly the SAM representation, particularly if the graph is arbitrarily structured. To manage this problem, I constructed the large scale simulation (see Section 3.4.6.2) which used randomly generated environments with randomly generated trajectories as a way to reduce the impact of exploration planning on the SAM result. With a large enough simulation, the performance results should converge to a coarse average case for evaluating that is independent of robot planning. It should be noted that these results could be improved by adapting the exploration algorithm to account for efficient exploration.

8.2 DDF-SAM Design Considerations

In order to actually apply DDF-SAM to a real-world multi-robot system, there are a number of design considerations that should be accounted for in order to achieve good performance. Before examining design decisions specific to DDF-SAM, I will discuss some general issues common to SAM systems.

The most significant assumption made in SAM approaches for perception is that the nonlinear optimization problem is convex. This appears in the SAM formulation as the assumption that measurements are corrupted with zero-mean, single-mode Gaussian noise. Even under this assumption, it is still possible to have local minima

in optimization. The most significant way in which this assumption can be violated is in outlier measurements, either due to unmodeled error, or a data association failure, which can substantially affect the solution of a smoothing least-squares formulation. In cases where robust data association is insufficient, it is possible to augment inference with either robust error models to minimize the effect of a spurious measurement, or to more actively model multi-modal distributions [119].

Another common assumption regarding acceptable online and realtime performance in SAM systems is the inherent growth of the underlying factor graph over time, which has implications on compute performance in both time and memory. I use as a benchmark the iSAM approach, which keeps update costs at a near constant level during normal operation, though for long run times or for large environments, the growth in costs may still be too much. There are a variety of long-term SLAM techniques that can be used to address this problem, such as by cutting out unnecessary factors and variables from the system through active sparsification [22, 23].

The following subsections examine how to engineer a DDF-SAM system, starting with making the best use of map summarization by determining which variables to share and how often to summarize, and then providing a design strategy for the system as a whole.

8.2.1 Which Variables to Share

The largest choice to make when applying DDF-SAM to a multi-robot application is which variables in a system should the robot share with its neighbors. I cast this as a trade-off between the utility of sending a particular set of variables vs. the additional cost incurred.

8.2.1.1 Evaluating Utility

The utility for each shared variable is substantially dependent on the objective for the full multi-robot team, as well as the capabilities of the robot platforms themselves.

Figure 61: Cooperative navigation example with quadrotors exploring an urban environment, in which one robot (highlighted in cyan) acts as a spotter for another robot (highlighted in magenta) that is flying close to a set of obstacles. In this case, the spotter can provide additional awareness outside of the sensor field of view of the exploring robot to ensure safety.

Some common criteria to evaluate are as follows:

- Commonly observable variables: If robots are sharing map information designed to facilitate localization within the map of a larger sensor area, it can be helpful for environmental variables to be commonly observable using the sensor systems on each platform. This can either derive from having the same sensors on each robot, or having an landmarks in the environment be observable across multiple sensor types, e.g., wall features that can be detected from both a planar laser scanner and from RGB-D cameras.

- Obstacles for navigation: When robots are cooperatively navigating an environment, it is useful for robots to share variables that represent obstacles that a robot should actively avoid. A useful case, illustrated in Figure 61, shows a "spotter" robot aiding a robot exploring a cluttered area. In this case, the spotter robot can share obstacle variables in such a way that the exploring robot can maneuver through more challenging openings.

- Robot poses: If robots directly observe each other, or move through common

environment that can facilitate scan-matching constraints, it will be necessary for robots to share poses from their trajectories. In the cooperative navigation scenario in Figure 61, the spotter robot can also provide estimates of the pose of exploring robot to further improve navigation precision.

- Number of shared variables: In most cases, the more variables shared between robots, the more detailed the fused maps can be. As the number of shared variables increases, the map quality will eventually converge on a full centralized solution.

8.2.1.2 Evaluating cost

While there are a variety of motivations for sharing variables between neighboring robots, there are costs that need to be addressed as well.

- More variables shared increase the communication cost: The size of messages between robots will increase with the number of variables, and for exact summarization, this cost is actually quadratic in the number of variables.

- Larger variables increase communication cost: As with increasing the number of variables, variables of higher dimension will incur additional cost.

- Data association metadata size: A hidden cost for differing variables is how much additional metadata, such as feature descriptors, is necessary for a receiving robot to perform multi-robot data association and incorporate a summarized map. For some cases, such scan-matching, it is necessary to transmit a full dense scan to a neighboring robot to allow for scan matching, which can be expensive.

In practice, there are some standard ways to handle this tradeoff between map utility and communication cost, as listed below:

- Share salient variables: a robot can determine which variables will be the most beneficial to neighboring robots, and only share a subset of those available. In

a scan-matching scenario, rather than sending all poses and scans to neighbors, the robot would only choose poses with scans that are likely to produce strong links with a neighboring robot.

- Use higher-order structure to create sparsely parametrized maps: This approach was used in the UPenn experiments (Section 4.5.2.2), in which wall features were used to replace dense scans of the environment for variables to send. The resulting scenario had only a small number of actual variables to represent the environment, which minimizes both communication and computation costs for DDF-SAM.

- Uncertainty on a variable: A robot may choose to only share variables that it has locally estimated to a particular level of certainty, which can help avoid sending variables that are poorly estimated locally that might cause data association failures on a receiving robot.

8.2.2 When to Summarize

Another key choice in applying DDF-SAM is determining when to share summarized maps and how often to perform summarization during online operation. The DDF-SAM opportunistic network model requires that robots their cached map information whenever neighboring robots come into communication range. However, it is not strictly necessary that each robot perform map summarization every time it comes into contact with a new neighboring robot. If a robot performs map summarization too frequently, the information gain between subsequent summarized maps can become too small to justify the cost. Not only is there a computational cost to performing map summarization on a given platform, a robot pushing new summarized maps out to the network frequently will incur additional communication costs, as well as induce more computation for receiving robots.

In this work, I have typically used heuristics to choose when robots perform summarization and share cached summarized map information with neighbors. Some examples of choices to make are:

- Fixed time interval: robots compute a new summarized map at a fixed interval of time t, ensuring that the most recent summarized map is never more than t timesteps old. This is the approach used in all of the results presented in this dissertation.

- Fixed distance interval: as the robots move through the environment, they summarize based on distance traveled, so that a stationary robot will not re-summarize its map unnecessarily.

- Fixed coverage interval: similar to the above approach, robots only summarize maps as they increase their coverage area, so that a robot not exploring new area will not re-summarize its map.

- Information gain: the most general approach would measure the amount of information gain since the last summarization occurred and only re-summarize after passing a threshold. This could account for both exploration that adds new variables, as well as summarizing after a large loop closure improved the certainty on the existing map.

Throughout this work, I have used synchronized summarization and sharing at fixed time intervals, in which every time the robots share summarized maps, they also compute a new summarized map.

8.2.3 Design Procedure

When integrating DDF-SAM in a multi-robot team with an existing single-robot SLAM approach, there are several useful steps for ensuring performance. The key requirements for the single-robot system are as follows

- Onboard sensing and computing suitable for performing SLAM.
- A local feature detection/data association algorithm suitable for recognizing known places and features without any significant outlier correspondences.
- A single-robot SLAM formulation using a SAM framework, incorporating necessary robot trajectory information as well as any constraints due to landmark observations or inter-pose matchings.

The key requirement for a single-robot SLAM formulation means that each robot should be able to perform local SLAM without any input from neighboring robots. This is necessary for two reasons: 1) ensuring that the robot can survive without communication with neighbors, and 2) ensuring that summarized maps shared with neighboring robots represent a locally optimal solution for the given measurements.

To evaluate whether the local SAM system is sufficient for ensuring convergence, it is easiest to evaluate the upper bound on map quality and convergence, which is a centralized batch solution using all of the available measurements. This is not designed to be representative of runtime performance, but rather whether it is possible to find a solution. The most representative centralized approach is to use a Gauss-Newton batch solver, which does not include any trust-region damping as in Levenberg-Marquardt, which would bias the convergence result. If there are not enough constraints, or the problem is ill-conditioned, the centralized solution will not converge, and it will be necessary to add more constraints, either on local maps, or between robots.

Once a centralized system can generate results, it is possible to choose the set of variables to share and how to perform summarization, as described in the previous sections. These decisions will need to account for the available communication in the robot network.

8.3 Overview

In this chapter, I provided an overview of how DDF-SAM and decentralized inference can fit into a multi-robot team scenario, starting with an analysis of the relationship between mission objectives and robot capabilities, and then analyzing DDF-SAM under these metrics. Because of the large variability between multi-robot coordination techniques, the evaluation focuses on an average case for exploration, designed to decouple the perception system from exploration. I then characterized the performance of DDF-SAM, particularly looking at scalability concerns.

In order to make clear some of the underlying assumptions of DDF-SAM, I presented an overview of design considerations applying decentralized inference to a variety of robot applications. In particular, I highlighted the choice how to share information between robots as a key determinant in how well the overall inference problem performs within a multi-robot team.

Chapter 9

DISCUSSION

In this dissertation, I introduced the DDF-SAM paradigm for fully decentralized smoothing and mapping in multi-robot systems that is online, scalable and consistent, by sharing and fusing summarized maps between neighboring robots. I have presented versions of the DDF-SAM approach, centering around using either batch nonlinear optimization to merge summarized maps, as in DDF-SAM 1.0, or using incremental solvers to merge both local map information and summarized maps, as in DDF-SAM 2.0. I developed and evaluated additional variations on DDF-SAM 2.0, to address the implications of separate linearization points between robots as well as extending summarization to include approximate techniques to trade-off map approximation for cheaper communication. I evaluated these approaches with a variety of real-world and simulated datasets to validate the intended goals.

9.1 Thesis

In this section, I will reproduce the thesis statement, claims and contributions and highlight how claims have been fulfilled throughout this work. The thesis statement for this work is as follows:

> *DDF-SAM provides a smoothing and mapping solution to inference in the decentralized data fusion problem that is online, scalable and consistent, while flexible enough to provide an extended sensor horizon under a variety of sensing modalities.*

The main evaluation criteria for this approach can be formulated as the following targeted characteristics for a solution to the multi-robot mapping problem:

Online DDF-SAM achieves online mapping both structurally and through performance metrics. Structurally, DDF-SAM is an online algorithm because the robots share data during operation and create maps at run-time. With regard to performance sufficient for online use, I have demonstrated experimentally the computational costs of operations in DDF-SAM are fast enough.

Scalable DDF-SAM primarily achieves scalability to increasing the number of robots in a system through the structural technique of fusion neighborhood bounding. In addition, the introduction of sparse approximate summarization techniques ensures a linear growth in communication bandwidth as robots explore.

Consistent The requirement within DDF-SAM that summarized maps share only locally observed information ensures I do not double-count information within cyclic networks. Furthermore, I evaluated the quality of the solutions for DDF-SAM for error, which were acceptably low in comparison to a pure local or centralized mapping solution.

Because DDF-SAM is a generic inference approach operating on sets of mutually observed variables between different robots, much of the goal of a system flexible enough for multiple sensing modalities is inherently solved, so long as the front-end system exists to initialize local SLAM.

As research contributions were spread across the chapters of this dissertation, I reiterate the contributions in reference to specific chapters in this document below:

- I introduced the DDF-SAM paradigm in Chapter 3, providing basis for the further claims in the paper, as well as an evaluation methodology. The main contributions from this section are map summarization, choosing shared variables, and the fusion neighborhood bound, all of which ensure scalability to a large team of robots.

- I developed DDF-SAM 1.0 [34, 35] in Chapter 4 as a full implementation of DDF-SAM centering around a batch nonlinear constrained optimization technique for fusing cached summarized maps. As a full implementation of the the approach, I conducted both real-world and simulated experiments [36] which demonstrate DDF-SAM produces maps with extended sensor horizons and operates at real time onboard a realistic robot platform. The real-world experiments [35, 72] included two types of sensors, with three different feature representations, validating claims of flexibility, and also introducing a multi-robot data association technique for matching constellation maps [35].

- I developed DDF-SAM 2.0 [32, 33] in Chapter 5 as a different approach to implementing DDF-SAM, this time targeting update performance for online operation by using incremental SAM (iSAM) [86, 85] techniques. The primary contributions from this work is the introduction of antifactor downdating to enforce consistency in summarized maps.

- I extended DDF-SAM 2.0 for full nonlinear systems in Chapter 6, introducing relinearization of previously linearized factors that allows for unknown global positions of robots. I evaluated this system in simulation, validating the claims it has online performance, faster than DDF-SAM 1.0, while maintaining the scalability and consistency characteristics of DDF-SAM.

- I focused on the operation of map summarization in Chapter 7, in which I developed and evaluated multiple approaches for exact and approximate map summarization techniques designed to trade-off between approximation and communication cost. I contributed both a naive Bayes approximate summarization technique and local tree approximate summarization, which drastically reduce the bandwidth cost of sharing summarized maps at the expense of additional computational cost and approximation.

9.2 Lessons Learned

While I have presented a variety of contributions to the state of the art, it important to note both assumptions and shortcomings in of this work, as they serve as a basis for future work. Many of these issues amount to simplifications designed to decouple other problems that should be relaxed in order to better integrate DDF-SAM into an autonomous or semi-autonomous multi-robot system.

The most significant assumption made during the course of this work is the decoupling of the data association problem from decentralized inference, at both local and global levels. This assumption removes a large class of failure modes common to least-squares methods that occur in the presence of measurement outliers, particularly incorrect feature correspondences, from the decentralized inference problem being studied which is a useful simplification for theoretical analysis, but for most real-world applications, data association is never perfect. For the real-world experiments I conducted, data association was a significant challenge, particularly to ensure both local and global consistency for feature labels, motivating the contribution of a multi-robot data association technique [35], though more work is necessary to demonstrably show consistency across a larger robot team.

The next significant assumption is the more subtle requirement on the mapping domain that the local map on each robot not be substantially encumbered by incorrect local minima. As single-robot SLAM is a nonlinear optimization operation, there is a possibility the estimation can become stuck in a poor local minima, which results in overly certain estimates around incorrect solutions. While even a local map stuck in a local minima can be summarized, a substantially wrong local minima could result in a bad map solution for the robot receiving the summarized map. In experiments, I circumvented this problem by ensuring each robot would only start sharing summarized maps after a fixed interval to let the map estimate converge locally.

One structural assumption made within DDF-SAM is it is possible to send a complete summarized maps every time communication occurs, in which I send what can be redundant information between robots. Sending a summarization of the whole local map to a robot that already has an older copy does introduce extra communication costs that could be avoided by sending updates to previous maps instead. I chose not to pursue this approach due to the substantially increased bookkeeping on each robot to distribute incremental updates to summarized maps. The result is that for severely restricted communication, as occurs with underwater acoustic modems over long range, it will likely be difficult to adapt existing DDF-SAM techniques.

While the system presented is not an end-to-end decentralized SLAM solution, I still make recommendations for general scenarios for which different variations on DDF-SAM are best suited. These are classified based on which robot platform capabilities are most restricted.

- Primarily communication-limited: Use nonlinear DDF-SAM 2.0 with local tree approximate summarization and a small value for K. This approach yields fast update times and a good approximation for low growth in the size of the messages over time.

- Severely communication-limited: The best available option is nonlinear DDF-SAM 2.0 with naive Bayes approximate summarization, with half the communication cost of tree approximation, with a small value for K.

- Applications needing high map precision for navigation: Use nonlinear DDF-SAM 2.0 with exact Bayes tree summarization with a large value of K to yield the most precise, detailed maps.

- Local mapping problems with local minima or outliers: Use DDF-SAM 1.0 using trust-region based batch solvers and robust error metrics for local solving, with a high value of K to provide more corrections from neighboring robots.

9.3 Future Work

Looking forward, there are a variety of interesting directions through which to expand this research, both to address the shortcomings described in the previous section and to extend the the capabilities.

Further exploiting decentralized collaboration between robots to minimize costs and create additional functional capabilities. Integrating data association and transform estimates into the sharing/caching formulation would be useful in scan-matching SLAM applications, where scans and the results of ICP can be distributed efficiently through the network without the need to compute ICP redundantly. To allow for more effective coupling with online cooperative navigation, robots could choose an *active set* of summarized maps based on a utility metric such that a robot would choose to maintain maps through areas of interest as a part of exploration. Finally, it would be possible to avoid excessive network transfer by choosing local information to share with neighbors, both from locally available maps and cached summarized maps, based on an information-gain metric for the robot network as a whole.

A remaining research question is to determine if there are more generalizable approaches for choosing the sharing of information between robots, particularly under constrained communication. Each robot r could solve a constrained optimization problem to determine what information to share with neighboring robot a, where the objective is to maximize the information gain of the robot a under the constraint of keeping the message size within a fixed bandwidth budget. One could imagine robot r could choose only those cached summarized maps to send to robot a that will fit within the bandwidth budget, or possibly even choose different set of shared variables or summarization technique. A more sophisticated approach could generalize beyond the pairwise information gain case, in which the robot r chooses to share information designed to maximize information gain to *the robot network as a whole*, where robot r might send might send more information to more highly connected neighboring robots

to ensure faster or more reliable propagation through the network. Facilitating this approach will likely require additional engineering of the approach robots use to share metadata before sending full summarized maps.

Another interesting area of research is to find an analytic formula that relates communication to map precision and accuracy, or at least act as a bound on expected performance. Within the networked systems literature there are a variety of analytic techniques for examining performance characteristics depending on the network connectivity. Extending these approaches to a decentralized least-squares solution as in the decentralized SAM case will likely be challenging due to how difficult it can be to make theoretical performance guarantees for arbitrary SAM graph structures. However, finding such an analytic solution or bound would make design of a decentralized mapping system easier to adapt to a given application. In addition, this could tie into the previously mentioned optimization of shared data, in which a robot r might not share summarized map data with a neighboring robot a if the mapping accuracy would not improve.

With regard to improving the performance of DDF-SAM, both in terms of computational costs and map quality, there are several avenues to explore. The benefits of using approximate summarization would be greater with a technique for enforcing consistency that is less overly conservative than the uniform covariance intersection weighting used in Chapter 7. While this might involve additional computation to optimize for better weights for variables, the improvement in precision will likely be worth the expense. The quality of map fusion could be improved by additional optimization of reference frame transformations using a graphical approach to ensure consistency between relative reference frames.

While this work presents experimental validation of the core claims, there is a need to integrate DDF-SAM into a realistic multi-robot system. While the UPenn experiments (Section 4.4.1) were performed online on a real robot platform, it would

be a good test of the system to use a microcontroller capable of deployment on the types of small flying robots envisioned for urban search and rescue applications.

9.4 Final Thoughts

As I have demonstrated throughout this dissertation, DDF-SAM is an effective approach for decentralized multi-robot SLAM that is online, scalable and consistent. This work should serve as a basis for further decentralized perception research and my hope is that it will have a positive impact towards autonomous multi-robot systems that can make their way into the field.

REFERENCES

[1] Absil, P.-A., Mahony, R., and Sepulchre, R., *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton University Press, 2007.

[2] Agarwal, P., Tipaldi, G., Spinello, L., Stachniss, C., and Burgard, W., "Robust map optimization using dynamic covariance scaling," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[3] Agarwal, P. and Olson, E., "Variable reordering strategies for SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, October 2012.

[4] Agarwal, S., Snavely, N., Seitz, S. M., and Szeliski, R., "Bundle adjustment in the large," in *European Conf. on Computer Vision (ECCV)*, 2010.

[5] Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R., "Building Rome in a day," in *Intl. Conf. on Computer Vision (ICCV)*, 2009.

[6] Andersson, L. and Nygards, J., "C-SAM : Multi-robot SLAM using square root information smoothing," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2008.

[7] Aragues, R., J.Cortes, and Sagues, C., "Distributed consensus on robot networks for dynamically merging feature-based maps," *IEEE Transactions on Robotics*, 2012.

[8] *Field Operations Guide, Urban Search and Rescue Structures Specialist*. U.S. Army Corp of Engineers, Urban Search and Rescue Program, DisasterEngineer, 7th ed., June 2013. Training documents used by FEMA.

[9] *Shoring Operations Guide*. U.S. Army Corp of Engineers, Urban Search and Rescue Program, DisasterEngineer, 3rd ed., June 2013. Training documents used by FEMA.

[10] Bahr, A., Walter, M., and Leonard, J., "Consistent cooperative localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3415–3422, May 2009.

[11] Bailey, T., Bryson, M., Mu, H., Vial, J., McCalman, L., and Durrant-Whyte, H., "Decentralised cooperative localisation for heterogeneous teams of mobile robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[12] Bailey, T. and Durrant-Whyte, H., "Simultaneous localisation and mapping (SLAM): Part II state of the art," *Robotics & Automation Magazine*, Sep 2006.

[13] Bailey, T., Julier, S., and Agamennoni, G., "On conservative fusion of information with unknown non-gaussian dependence," in *Intl. Conf. on Information Fusion, FUSION*, 2012.

[14] Bar-Shalom, Y. and Li, X., *Estimation and Tracking: principles, techniques and software*. Boston, London: Artech House, 1993.

[15] Barfoot, T., "Online visual motion estimation using FastSLAM with sift features," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 579–585, Aug 2005.

[16] Bay, H., Tuytelaars, T., and Gool, L. V., "Surf: speeded up robust features," in *European Conf. on Computer Vision (ECCV)*, 2006.

[17] Bierman, G., "An application of the square-root information filter to large scale linear interconnected systems," *IEEE Trans. Automat. Contr.*, vol. 23, pp. 91–93, February 1978.

[18] Bosse, M., Newman, P., Leonard, J., and Teller, S., "Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework," *Intl. J. of Robotics Research*, vol. 23, pp. 1113–1139, Dec 2004.

[19] Bryson, M. and Sukkarieh, S., "Architectures for cooperative airborne simultaneous localisation and mapping," *Journal of Intelligent and Robotic Systems, Special Issue on Airborne SLAM*, vol. 55, pp. 267–297, June 2009.

[20] Bui, M., Butelle, F., and Lavault, C., "A distributed algorithm for constructing a minimum diameter spanning tree," *J. of Parallel and Distributed Computing*, 2004.

[21] Carlevaris-Bianco, N. and Eustice, R. M., "Generic factor-based node marginalization and edge sparsification for pose-graph SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 5728–5735, 2013.

[22] Carlevaris-Bianco, N. and Eustice, R. M., "Long-term simultaneous localization and mapping with generic linear constraint node removal," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, (Tokyo, Japan), November 2013.

[23] Carlevaris-Bianco, N. and Eustice, R. M., "Conservative edge sparsification for graph SLAM node removal," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Hong Kong, China), June 2014. Accepted, To Appear.

[24] Carlone, L., Ng, M. K., Du, J., Bona, B., and Indri, M., "Rao-Blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 243–249, 2010.

[25] Castellanos, J., MartÏ¿œnez-CantÏ¿œn, R., TardÏ¿œs, J., and J., N., "Robocentric map joining: Improving the consistency of EKF-SLAM," *Robotics and Autonomous Systems*, vol. 55, pp. 21–29, January 2007.

[26] Censi, A., "An ICP variant using a point-to-line metric," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Pasadena, CA), May 2008.

[27] Censi, A., "On achievable accuracy for pose tracking," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Kobe, Japan), May 2009.

[28] Chapman, A. and Sukkarieh, S., "A protocol for decentralized multi-vehicle mapping with limited communication connectivity," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 357–362, 2009.

[29] Chli, M., *Applying Information Theory to Efficient SLAM*. PhD thesis, Imperial College London, October 2009.

[30] Chong, K. and Kleeman, L., "Large scale sonarray mapping using multiple connected local maps," in *Field and Service Robotics (FSR)*, 1997.

[31] Chow, C. and Liu, C., "Approximating discrete probability distributions with dependence trees," *IEEE Trans. Inform. Theory*, vol. 14, pp. 462–467, May 1968.

[32] Cunningham, A., Indelman, V., and Dellaert, F., "Consistent decentralized graphical SLAM with anti-factor down-dating," in *IEEE Intl. Conf. on Safety, Security and Rescue Robotics (SSRR)*, (College Station, TX.), November 2012. accepted as late-breaking report.

[33] Cunningham, A., Indelman, V., and Dellaert, F., "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), May 2013.

[34] Cunningham, A., Paluri, M., and Dellaert, F., "DDF-SAM: Fully distributed slam using constrained factor graphs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[35] Cunningham, A., Wurm, K., Burgard, W., and Dellaert, F., "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (St. Paul, MN), 2012.

[36] Cunningham, A. and Dellaert, F., "Large-scale experimental design for decentralized SLAM," in *Unmanned Systems Technology XIV, SPIE Defense, Security, and Sensing*, (Baltimore, MD), 2012.

[37] Davison, A., "Real-time simultaneous localisation and mapping with a single camera," in *Intl. Conf. on Computer Vision (ICCV)*, pp. 1403–1410, Oct 2003.

[38] Dellaert, F., "Addressing the correspondence problem: A Markov chain Monte Carlo approach," Tech. Rep. CMU-RI-TR-00-11, Robotics Institute, School of Computer Science, Carnegie Mellon, January 2000.

[39] Dellaert, F., *Monte Carlo EM for Data Association and its Applications in Computer Vision*. PhD thesis, School of Computer Science, Carnegie Mellon, September 2001. Also available as Technical Report CMU-CS-01-153.

[40] Dellaert, F., "Square Root SAM: Simultaneous location and mapping via square root information smoothing," in *Robotics: Science and Systems (RSS)*, 2005.

[41] Dellaert, F., Carlson, J., Ila, V., Ni, K., and Thorpe, C., "Subgraph-preconditioned conjugate gradient for large scale slam," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[42] Dellaert, F. and Kaess, M., "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, pp. 1181–1203, Dec 2006.

[43] Dellaert, F., Seitz, S., Thorpe, C., and Thrun, S., "Structure from motion without correspondence," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2000.

[44] Dellaert, F., Seitz, S., Thorpe, C., and Thrun, S., "Feature correspondence: A Markov chain Monte Carlo approach," in *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, 2001.

[45] Djugash, J. and Singh, S., "Motion-aided network slam with range," *Intl. J. of Robotics Research*, vol. 31, pp. 603 – 625, April 2012.

[46] Djugash, J., Singh, S., and Grocholsky, B. P., "Decentralized mapping of robot-aided sensor networks," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 583–589, May 2008.

[47] Do Carmo, M. P., *Riemannian geometry*. Springer, 3rd ed., 1992.

[48] Do Carmo, M. P., *Differential geometry of curves and surfaces*, vol. 2. Prentice-Hall Englewood Cliffs, 1976.

[49] Durrant-Whyte, H. and Stevens, M., "Data fusion in decentralized sensing networks," in *4th Intl. Conf. on Information Fusion*, 2001.

[50] Durrant-Whyte, H. F., Rao, B. Y. S., and Hu, H., "Toward a fully decentralized architecture for multi-sensor data fusion," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, pp. 1331–1336, 1990.

[51] Durrant-Whyte, H. and Bailey, T., "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics & Automation Magazine*, Jun 2006.

[52] Eustice, R., Singh, H., and Leonard, J., "Exactly sparse delayed-state filters," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 2417–2424, April 2005.

[53] Eustice, R., Walter, M., and Leonard, J., "Sparse extended information filters: Insights into sparsification," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3281–3288, Aug 2005.

[54] Fiala, M., "Artag, a fiducial marker system using digital techniques," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[55] Fielding, G. and Kam, M., "Applying the Hungarian method to stereo matching," in *IEEE Conference on Decision and Control*, pp. 549–558, December 1997.

[56] Fischler, M. and Bolles, R., "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[57] Folkesson, J., Jensfelt, P., and Christensen, H., "Graphical SLAM using vision and the measurement subspace," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Aug 2005.

[58] Folkesson, J., Jensfelt, P., and Christensen, H., "Vision SLAM in the measurement subspace," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Apr 2005.

[59] Folkesson, J., *Simultaneous localization and mapping with robots*. PhD thesis, KTH, Numerical Analysis and Computer Science, NADA, 2005.

[60] Fox, D., Burgard, W., Kruppa, H., and Thrun, S., "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[61] Grisetti, G., Stachniss, C., and Burgard, W., "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 667–672, 2005.

[62] Grisetti, G., Stachniss, C., and Burgard, W., "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[63] Grisetti, G., Stachniss, C., and Burgard, W., "Non-linear constraint network optimization for efficient map learning," *Trans. on Intelligent Transportation systems*, 2009.

[64] Grisetti, G., Stachniss, C., Grzonka, S., and Burgard, W., "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Robotics: Science and Systems (RSS)*, Jun 2007.

[65] Guivant, J., Nebot, E., Nieto, J., and Masson, F., "Navigation and mapping in large unstructured environments," *Intl. J. of Robotics Research*, vol. 23, pp. 449–472, April 2004.

[66] Harris, C. and Stephens, M., "A combined corner and edge detector," *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, August 1988.

[67] Hartley, R. I. and Zisserman, A., *Multiple View Geometry in Computer Vision*. Cambridge University Press, second ed., 2004.

[68] Howard, A., "Multi-robot mapping using manifold representations," in *IEEE International Conference on Robotics and Automation*, (New Orleans, Louisiana), pp. 4198–4203, Apr 2004.

[69] Howard, A., "Multi-robot simultaneous localization and mapping using particle filters," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[70] Howard, A., Sukhatme, G. S., and Matarić, M. J., "Multi-robot mapping using manifold representations," *Proceedings of the IEEE - Special Issue on Multi-robot Systems*, vol. 94, pp. 1360 – 1369, Jul 2006.

[71] Huber, P., *Robust Statistics*. John Wiley & Sons, New York, NY, 1981.

[72] III, J. G. R., Cunningham, A., Paluri, M., Christensen, H. I., Dellaert, F., Michael, N., Kumar, V., and Mathies, L., "Results of mast joint experiment 3.1," in *Defense, Security and Sensing*, (Orando, Fl.), SPIE, April 2011.

[73] III, J. G. R., Trevor, A. J. B., Nieto-Granda, C., Cunningham, A., Paluri, M., Michael, N., and Christensen, H. I., "Effects of sensory precision on mobile robot localization and mapping," in *Intl. Sym. on Experimental Robotics (ISER)*, (Delhi, India), IFRR, Dec 2010.

[74] Indelman, V., Gurfil, P., Rivlin, E., and Rotstein, H., "Graph-based distributed cooperative navigation for a general multi-robot measurement model," *Intl. J. of Robotics Research*, vol. 31, August 2012.

[75] Jian, Y.-D., Balcan, D., and Dellaert, F., "Generalized subgraph preconditioners for large-scale bundle adjustment," in *Intl. Conf. on Computer Vision (ICCV)*, 2011.

[76] Julier, S. J., Bailey, T., and Uhlmann, J. K., "Using exponential mixture models for suboptimal distributed data fusion," in *Nonlinear Statistical Signal Processing Workshop*, pp. 160–163, IEEE, 2006.

[77] Julier, S. J., Jeffrey, and Uhlmann, K., "Unscented filtering and nonlinear estimation," in *Proceedings of the IEEE*, pp. 401–422, 2004.

[78] Julier, S. and Uhlmann, J. K., "A new extension of the kalman filter to nonlinear systems," in *AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, (Orlando, FL), SPIE, 1997.

[79] Julier, S. and Uhlmann, J. K., "Using covariance intersection for slam," *Robotics and Autonomous Systems*, vol. 55, pp. 3–20, Jan. 2007.

[80] Julier, S. and Uhlmann, J., "A non-divergent estimation algorithm in the presence of unknown correlations," in *American Control Conference*, pp. 2369–73, 1997.

[81] Julier, S. and Uhlmann, J., "A counter example to the theory of simultaneous localization and map building," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 4, pp. 4238–4243, 2001.

[82] Julier, S. and Uhlmann, J., "General decentralized data fusion with covariance intersection (CI)," in *Handbook of Data Fusion* (Hall, D. and Llinas, J., eds.), ch. 13, Boca Raton FL, USA: CRC Press, 2001.

[83] Kaess, M., Ila, V., Roberts, R., and Dellaert, F., "The Bayes tree: An algorithmic foundation for probabilistic robot mapping," in *Intl. Workshop on the Algorithmic Foundations of Robotics*, Dec 2010.

[84] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F., "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Shanghai, China), May 2011.

[85] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F., "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, pp. 217–236, Feb 2012.

[86] Kaess, M., Ranganathan, A., and Dellaert, F., "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, pp. 1365–1378, Dec 2008.

[87] Kalman, R. E., "A new approach to linear filtering and prediction problems," *Trans. ASME, Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[88] Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., and Teller, S., "Multiple relative pose graphs for robust cooperative mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Anchorage, Alaska), pp. 3185–3192, May 2010.

[89] Koller, D. and Friedman, N., *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[90] Konolige, K. and Agrawal, M., "FrameSLAM: from bundle adjustment to realtime visual mapping," *IEEE Trans. Robotics*, vol. 24, no. 5, pp. 1066–1077, 2008.

[91] Kschischang, F., Frey, B., and Loeliger, H.-A., "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, February 2001.

[92] Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., and Kleiner, A., "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009.

[93] Lindstrom, P. and Wedino, P., "Gauss-newton based algorithms for constrained nonlinear least squares problems," Tech. Rep. UMINF-901.87, Institute of Information Processing, University of Umea, Sweden, 1988.

[94] Lowe, D., "Distinctive image features from scale-invariant keypoints," *Intl. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[95] Lu, F. and Milios, E., "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, pp. 333–349, Apr 1997.

[96] Lu, F. and Milios, E., "Robot pose estimation in unknown environments by matching 2D range scans," *J. of Intelligent and Robotic Systems*, p. 249:275, April 1997.

[97] MacKay, D., *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[98] Makarenko, A., Brooks, A., Durrant-Whyte, H., and Dellaert, F., "Decentralised data fusion: A graphical model approach," in *Proc. Intl. Conf. on Information Fusion*, 2009.

[99] Makarenko, A., Brooks, A., Durrant-Whyte, H., and Dellaert, F., "Decentralised data fusion: A graphical model approach," in *Proc. Intl. Conf. on Information Fusion*, 2009.

[100] Makarenko, A. and H, D.-W., "Decentralized data fusion and control in active sensor networks," in *7th Intl. Conf. on Information Fusion (Fusion'04)*, (Stockholm), June-July 2004.

[101] Mesbahi, M. and Egerstedt, M., *Graph Theoretic Methods for Multiagent Networks*. Princeton University Press, 2010.

[102] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B., "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc.* 19^{th} *AAAI National Conference on AI*, (Edmonton, Alberta, Canada), 2002.

[103] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B., "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Intl. Joint Conf. on AI (IJCAI)*, 2003.

[104] Murphy, R. R., Pratt, K. S., and Burke, J. L., "Crew roles and operational protocols for rotary-wing micro-uavs in close urban environments," in *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, (New York, NY, USA), pp. 73–80, ACM, 2008.

[105] Murphy, R. R., Steimle, E., Griffin, C., Cullins, C., Hall, M., and Pratt, K., "Cooperative use of unmanned sea surface and micro aerial vehicles at hurricane wilma," *J. Field Robot.*, vol. 25, no. 3, pp. 164–180, 2008.

[106] Murphy, R. R. and Stover, S., "Rescue robots for mudslides: A descriptive study of the 2005 la conchita mudslide response: Field reports," *J. Field Robot.*, vol. 25, no. 1-2, pp. 3–16, 2008.

[107] Murphy, R. R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., and Erkmen, A. M., "Search and rescue robotics," in *Springer Handbook of Robotics* (Siciliano, B. and Khatib, O., eds.), pp. 1151–1171, Springer Verlag, 2008.

[108] Murphy, R., "Human-robot interaction in rescue robotics," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, pp. 138 –153, may 2004.

[109] Neira, J. and Tardos, J., "Data association in stochastic mapping using the joint compatibility test," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 890–897, Dec. 2001.

[110] Nerurkar, E., Roumeliotis, S., and Martinelli, A., "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1402–1409, May 2009.

[111] Nettleton, E., Thrun, S., Durrant-Whyte, H., and Sukkarieh, S., "Decentralised SLAM with low-bandwidth communication for teams of vehicles," in *Field and Service Robotics (FSR)*, (Japan), July 2003.

[112] Ni, K., Jin, H., and Dellaert, F., "Groupsac: Efficient consensus in the presence of groupings," in *Intl. Conf. on Computer Vision (ICCV)*, (Kyoto; Japan), October 2009.

[113] Ni, K. and Dellaert, F., "Multi-level submap based slam using nested dissection," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[114] Nüchter, A., *3D robotic mapping: the simultaneous localization and mapping problem with six degrees of freedom*, vol. 52. Springer, 2009.

[115] Nüchter, A., Lingemann, K., Hertzberg, J., and Surmann, H., "6D SLAM-3D Mapping Outdoor Environments," *J. of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[116] Olson, E., Leonard, J., and Teller, S., "Fast iterative alignment of pose graphs with poor initial estimates," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 2262–2269, May 2006.

[117] Olson, E., *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.

[118] Olson, E., "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[119] Olson, E. and Agarwal, P., "Inference on networks of mixtures for robust robot mapping," *Intl. J. of Robotics Research*, vol. 32, no. 7, pp. 826–840, 2013.

[120] Olson, E., Strom, J., Morton, R., Richardson, A., Ranganathan, P., Goeddel, R., Bulic, M., Crossman, J., and Marinier, B., "Progress towards multi-robot reconnaissance and the MAGIC 2010 competition," *J. of Field Robotics*, September 2012.

[121] Pfingsthorn, M., Slamet, B., and Visser, A., "A scalable hybrid multi-robot SLAM method for highly detailed maps," in *RoboCup 2007: Robot Soccer World Cup XI* (Visser, U., Ribeiro, F., Ohashi, T., and Dellaert, F., eds.), Lecture Notes In AI, pp. 457–464, Springer-Verlag, July 2008.

[122] Pratt, K. S., Murphy, R., Stover, S., and Griffin, C., "Conops and autonomy recommendations for vtol small unmanned aerial system based on hurricane katrina operations," *J. Field Robot.*, vol. 26, no. 8, pp. 636–650, 2009.

[123] Pratt, K., Murphy, R., Burke, J., Craighead, J., Griffin, C., and Stover, S., "Use of tethered small unmanned aerial system at berkman plaza ii collapse," in *Safety, Security and Rescue Robotics, 2008. SSRR 2008. IEEE International Workshop on*, pp. 134 –139, oct. 2008.

[124] Ranganathan, P., Morton, R., Richardson, A., Strom, J., Goeddel, R., Bulic, M., and Olson, E., "Coordinating a team of robots for urban reconnaisance," in *Proceedings of the Land Warfare Conference (LWC)*, November 2010.

[125] Rodriguez-Losada, D., Matia, F., and Jimenez, A., "Local maps fusion for real time multirobot indoor simultaneous localization and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, pp. 1308–1313, 2004.

[126] Rosen, D., Kaess, M., and Leonard, J., "Robust incremental online inference over sparse factor graphs: Beyond the Gaussian case," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), May 2013.

[127] Roumeliotis, S. I. and Bekey, G. A., "Synergetic localization for groups of mobile robots," in *IEEE Conference on Decision and Control*, vol. 4, pp. 3477–3482, 2000.

[128] Roumeliotis, S. and Bekey, G., "Distributed multi-robot localization," *IEEE Trans. Robot. Automat.*, August 2002.

[129] Scaramuzza, D., "1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *Intl. J. of Computer Vision*, pp. 1–12, 2011.

[130] Shi, J. and Tomasi, C., "Good features to track," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, 1994.

[131] Smith, R. and Cheeseman, P., "On the representation and estimation of spatial uncertainty," *Intl. J. of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1987.

[132] Snavely, N., Seitz, S., and Szeliski, R., "Photo tourism: Exploring photo collections in 3D," in *SIGGRAPH*, pp. 835–846, 2006.

[133] Solà, J., Vidal-Calleja, T., Civera, J., and Montiel, J. M. M., "Impact of landmark parametrization on monocular EKF-SLAM with points and lines," *Intl. J. of Computer Vision*, September 2011.

[134] Stroup, A. W. and Balch, T., "Value-based action selection for observation with robot teams using probabilistic techniques," *Robotics and Autonomous Systems*, vol. 50, pp. 85–97, Feb. 2005.

[135] Stroupe, A., Martin, M., and Balch, T., "Distributed sensor fusion for object position estimation by multi-robot systems," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, pp. 1092–1098 vol.2, 2001.

[136] Tardï¿œs, J., Neira, J., Newman, P., and Leonard, J., "Robust mapping and localization in indoor environments using sonar data," *Intl. J. of Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[137] Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[138] Thrun, S. and Liu, Y., "Multi-robot SLAM with sparse extended information filters," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, (Sienna, Italy), Springer, 2003.

[139] Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., and Durrant-Whyte, H., "Simultaneous localization and mapping with sparse extended information filters," *Intl. J. of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[140] Thrun, S., Martin, C., Liu, Y., Hähnel, D., Emery-Montemerlo, R., Chakrabarti, D., and Burgard, W., "A real-time expectation maximization algorithm for acquiring multi-planar maps of indoor environments with mobile robots," *IEEE Trans. Robot. Automat.*, vol. 20, pp. 433–443, June 2004.

[141] Trevor, A. J. B., Rogers III, J. G., and Christensen, H. I., "Planar surface slam with 3d and 2d sensors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (St. Paul, MN), IEEE, May 2012.

[142] Vial, J., Durrant-Whyte, H., and Bailey, T., "Conservative sparsification for efficient and consistent approximate estimation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 886–893, IEEE, 2011.

[143] Wagner, D. and Schmalsteig, D., "Artoolkitplus for pose tracking on mobile devices," in *Computer Vision Winter Workshop*, 2007.

[144] Walter, M. R., Eustice, R. M., and Leonard, J. J., "Exactly sparse extended information filters for feature-based SLAM," *Intl. J. of Robotics Research*, vol. 26, pp. 335–359, April 2007.

[145] Wang, C., H., H. S., Yada, S., and Rosenfeld, A., "Some experiments in relaxation image matching using corner features," *Pattern Recognition*, vol. 16, no. 2, pp. 167–182, 1983.

[146] Wang, Z., *Exactly Sparse Information Filters for Simultaneous Localization and Mapping*. PhD thesis, The University of Technology, Sydney, 2007.

[147] Webster, S. E., Whitcomb, L. L., and Eustice, R. M., "Preliminary results in decentralized estimation for single-beacon acoustic underwater navigation," in *Robotics: Science and Systems (RSS)*, (Zaragoza, Spain), June 2010.

[148] Williams, S., Dissanayake, G., and Durrant-Whyte, H., "Towards multi-vehicle simultaneous localisation and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2002.